



UNIVERSITÀ DEGLI STUDI
DI TRENTO



AEI, Hannover

Leibniz Universität Hannover

What is LTPDA?

M Hewitson for the LTP Team

NPL

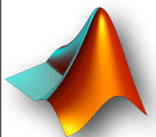
9th August 2010



Outline



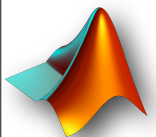
- Why LTPDA?
- Formalities
- Installation
- Topic 1: Introducing LTPDA - the basics
- Topic 2: Preprocessing data
- Topic 3: Spectral analysis
- Topic 4: Transfer functions and digital filters
- Topic 5: Fitting data



Who he?



- Born in Carlisle
- First 8 years of my working life as electrician
- MSci and PhD from University of Glasgow
 - Thesis: *“On aspects of characterising and calibrating the interferometric gravitational wave detector, GEO 600”*
- 7 years building, commissioning and calibrating GEO600
- Past 3 years as head of LTP Data Analysis team
- Married, two you children



The LTPDA team...



Martin Hewitson
Miquel Nofrarias
Ingo Diepholz
Anneke Monsky
Heather Audley



UNIVERSITÀ DEGLI STUDI
DI TRENTO

Mauro Hueller
Luigi Ferraioli
Giuseppe Congedo
Fabrizio De Marchi



Andrea Manica
Andrea Mattioli



Eric Plagniol



Adrien Grynagier

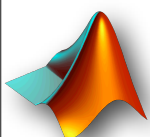


Michele Armano

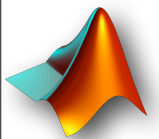


Marc Diaz Aguiló

LTPDA Introduction, NPL, 9th August 2010

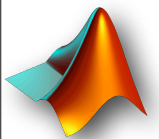


Not a commercial product!



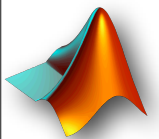
Not a commercial product!

- We have about 2 FTEs currently working on the toolbox



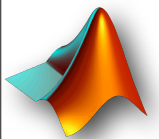
Not a commercial product!

- We have about 2 FTEs currently working on the toolbox



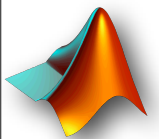
Not a commercial product!

- We have about 2 FTEs currently working on the toolbox
- Total effort so far is about 10 FTE years



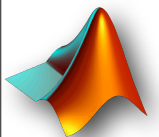
Not a commercial product!

- We have about 2 FTEs currently working on the toolbox
- Total effort so far is about 10 FTE years



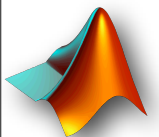
Not a commercial product!

- We have about 2 FTEs currently working on the toolbox
- Total effort so far is about 10 FTE years
- Feedback is essential



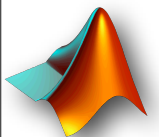
Not a commercial product!

- We have about 2 FTEs currently working on the toolbox
- Total effort so far is about 10 FTE years
- Feedback is essential



Not a commercial product!

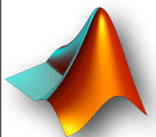
- We have about 2 FTEs currently working on the toolbox
- Total effort so far is about 10 FTE years
- Feedback is essential
- We will help whenever we can



Why?



- Data analysis for LISA Pathfinder mission will be done on-line
 - allow re-planning of upcoming experiments and investigations
- Front-line analysis done in STOC
 - pre-planned data analysis pipelines for each experiment of the mission
- Off-line analysis
 - follow-up, problem solving, etc
- Results must have a long shelf-life (up to LISA commissioning)
- Requirements:
 - flexible and robust data analysis environment
 - graphical user interface (for non-programming-experts)
 - high-level of testing
 - automated capture of processing chain
 - data doesn't exist in isolation



Formalities



- LTPDA:
 - <http://www.lisa.aei-hannover.de/ltpda/>
- Bugs and features:
 - <https://ed.fbk.eu/ltpda/mantis/>

LTPDA


a MATLAB® toolbox for accountable and reproducible data analysis

LTPDA is a MATLAB toolbox that uses an object-oriented approach to data analysis. LTPDA Objects are processed through a data analysis pipeline. At each analysis step, a record is kept of exactly what algorithm was applied to which object and with which parameters. In this way, the result of a particular data analysis is one or more objects, each containing the final result as numerical data together with a full processing history of how the result was achieved.


Latest version: **V2.3**

LTPDA includes algorithms and objects for

1. pre-processing of time-series data
2. performing spectral analysis of various kinds
3. performing digital filtering via IIR and FIR filters
4. constructing pole/zero models
5. constructing state-space models
6. and much more



- home
- Installation
- System requirements
- Downloads
- File repository
- Release Schedule
- User manual
- Training Sessions
- Documents
- Bugs and features
- Troubleshooting
- LTPDA Repository



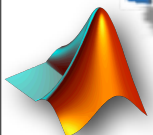
logged in as: martinheutson (Martin heutson - developer) 2010-07-30 09:07 CEST Project: All Projects Switch

Main | My View | View Issues | Report Issue | Change Log | Roadmap | Docs | My Account | Plugins | Logout

Search: Apply Filter [Advanced Filters] [Create Permalink] [Reset Filter] [Save Current Filter]

Viewing Issues (1 - 50 / 199) [Print Reports] [Graph] [CSV Export] [First Prev 1 2 3 4 Next Last]

P	ID	#	Category	Severity	Status	Updated	Summary
<input type="checkbox"/>		0000169	[DA_LTPDA_WORKBENCH] Bug Report	minor	acknowledged	2010-07-28	Deleting a global parameter (Constant Block) after execution cause that the value persist into the Matlab Workspace
<input type="checkbox"/>		0000403	[DA_LTPDA_TOOLBOX] Bug Report	minor	new	2010-07-28	Plot of different units
<input type="checkbox"/>		0000242	[DA_LTPDA] Bug Report	minor	feedback (huebler)	2010-07-28	Summing objects with different x base
<input type="checkbox"/>		0000494	[DA_LTPDA_TOOLBOX] Change Request	minor	resolved (martinheutson)	2010-07-28	A method to inherit (from another method) one parameter
<input type="checkbox"/>		0000495	[DA_LTPDA_TOOLBOX] Bug Report	major	new	2010-07-22	ltpda_filter/impresp has fixed sample rate
<input type="checkbox"/>		0000419	[DA_LTPDA_TOOLBOX] Bug Report	text	assigned (huebler)	2010-07-11	Typo: wrong XUNITS value for Time-series Plot constructor
<input type="checkbox"/>		0000472	[DA_LTPDA_TOOLBOX] Bug Report	minor	assigned (huebler)	2010-07-11	Exponents in units are badly shown in x-axes
<input type="checkbox"/>		0000493	[DA_LTPDA] Change Request	feature	new	2010-06-25	Load objects from multiple files.



Mailing lists



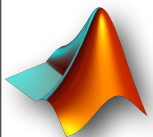
- ltnda_releases@aei.mpg.de
- http://lists.aei.mpg.de/cgi-bin/mailman/listinfo/ltnda_releases
 - for releases of LTPDA

- ltnda_users@aei.mpg.de
- http://lists.aei.mpg.de/cgi-bin/mailman/listinfo/ltnda_users
 - for LTPDA users

- ltnda_dev@aei.mpg.de
- http://lists.aei.mpg.de/cgi-bin/mailman/listinfo/ltnda_dev
 - for core development team

- ltnda_cvs@aei.mpg.de
- http://lists.aei.mpg.de/cgi-bin/mailman/listinfo/ltnda_cvs
 - for LTPDA CVS commit mails

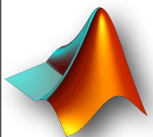
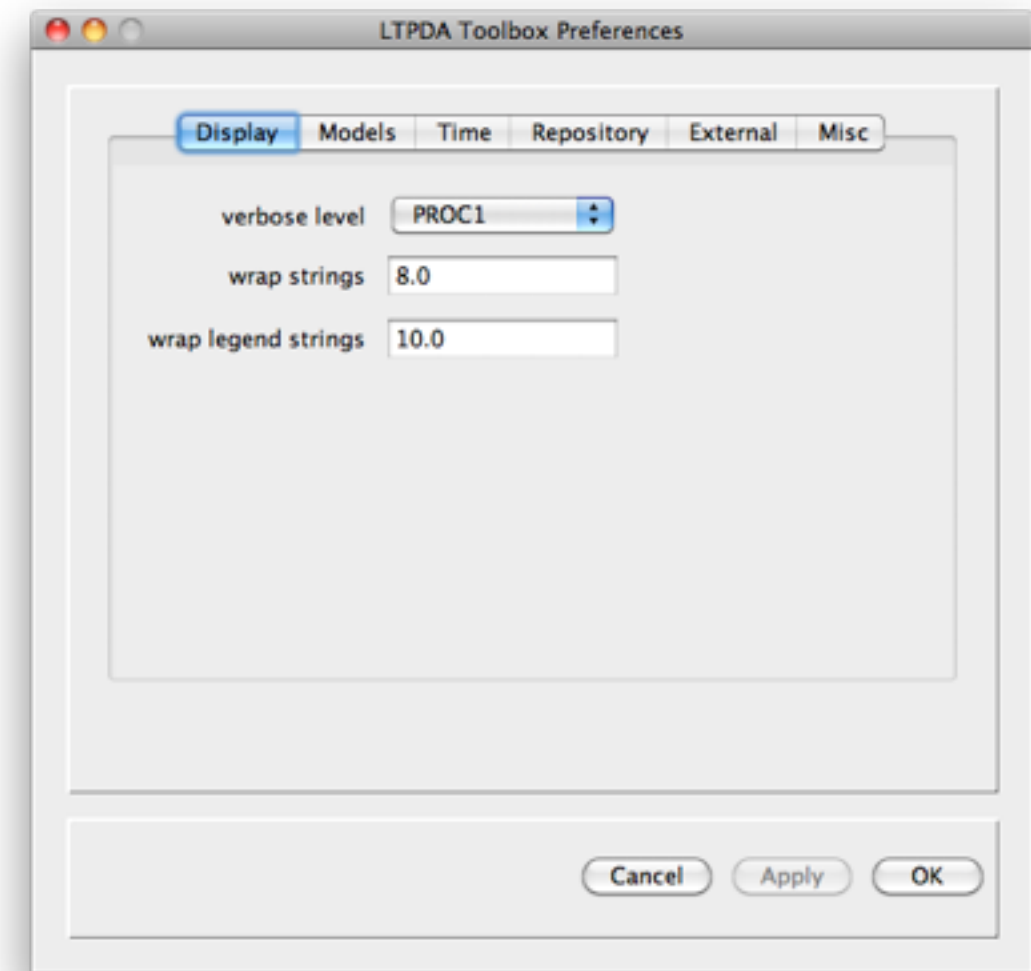
- ltnda_meeting@aei.mpg.de
- http://lists.aei.mpg.de/cgi-bin/mailman/listinfo/ltnda_meeting
 - for meeting announcements



Installation



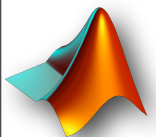
1. Download latest version from
1.1. <http://www.lisa.aei-hannover.de/ltpda/>
2. Unzip to somewhere
3. Start MATLAB
4. File->Set Path...
5. Click 'Add with Subfolders...'
6. Navigate to the ltpda_toolbox folder
you unzipped
7. On MATLAB terminal
 - 7.1. `>> ltpda_startup`
 - 7.1.1. (you can add this command to you
normal startup.m file)
8. Set preferences
 - 8.1. for now just click 'Apply' and close the
GUI



Post-installation steps

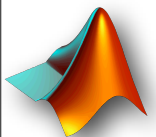
- Test installation
 - >> run_tests
- Install graphviz
 - see LTPDA user manual (>> doc)
 - LTPDA Toolbox
 - Getting Started with the LTPDA Toolbox
 - Additional 3rd-party software

http://www.lisa.aei-hannover.de/ltlda/usermanual/ug/additional_progs.html



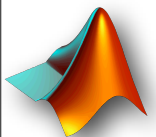
Updating the toolbox

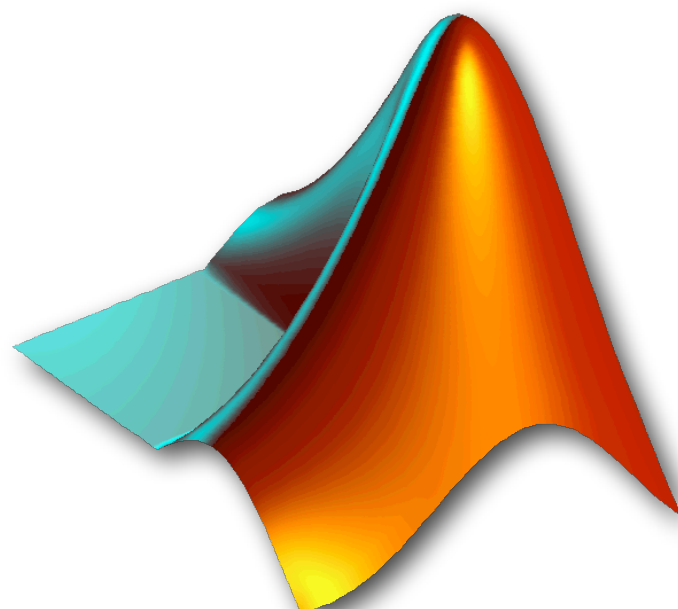
- Remove old toolbox from MATLAB path
 - File -> Set Path
 - Select all in list containing 'ltpda_toolbox'
 - Click 'Remove'
- Install new toolbox as per previous instructions



Submitting a bug or feature request

- Go to <https://ed.fbk.eu/ltpda/mantis/>
- Self-sign-up
 - click 'Signup for a new account'
- Once your account is active, you can log in
- To report an issue:
 - First search the existing issues in case your problem is covered!
 - Click 'Report Issue'
 - Choose project 'DA_LTPDA'
 - Select 'bug report' or 'change request'
 - Complete the form with as much information as possible





Topic 1

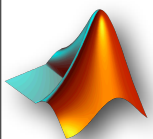


- The basics
 - Object-oriented programming (for beginners)
 - Analysis Objects
 - How history tracking works
 - Other objects
 - Parameter lists
 - Building objects
 - Setting object properties
 - Viewing history
 - Making time-series AOs
 - Basic math
 - Saving and loading
 - Reading data files
 - Writing LTPDA scripts
- Hands-on

```
****
**
-----
  ///  /  \\\
 ///  /  \\\
** | +---+ / +---+ | **
*** | | //-----| ***
** | +---+ / +---+ | **
  \\\  /  ///
   \\\  ///
-----
      **
      ****

Welcome to LTPDA Toolbox

Version: 2.3.1
Release: (R2010a)
Date: 30-07-10
```

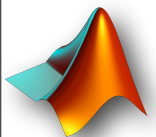


Object-oriented programming

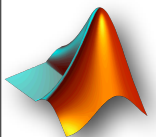


- Learn these words:

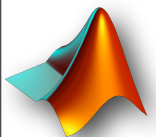
- class
- object
- instance
- method
- constructor
- property
- inheritance



- A class is a description of an object
- Examples:
 - aeroplane, house, vehicle, animal
 - algorithm, colour, sentence



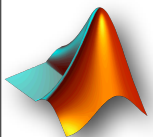
- A class is a description of an object
- Examples:
 - aeroplane, house, vehicle, animal
 - algorithm, colour, sentence



Object, instance



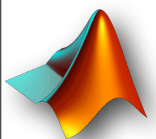
- An object is an instance of a class
- Examples:
 - Airforce 1 (*aeroplane*), me (*person*), garfield (*cartoon cat*)
 - mean (*algorithm*), red (*color*), “isn’t this easy?” (*sentence*)



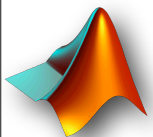
Object, instance



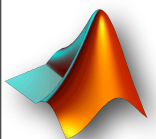
- An object is an instance of a class
- Examples:
 - Airforce 1 (*aeroplane*), me (*person*), garfield (*cartoon cat*)
 - mean (*algorithm*), red (*color*), “isn’t this easy?” (*sentence*)



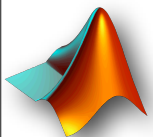
- Something that acts on an object (instance of a class)
- Examples:
 - 'start' a 'car' - 'start' is a method of the class 'car'
 - 'start' my car - use the method 'fly' on my car
 - mean(x) - use the method 'mean' on the data object 'x'



- Something that acts on an object (instance of a class)
- Examples:
 - 'start' a 'car' - 'start' is a method of the class 'car'
 - 'start' my car - use the method 'fly' on my car
 - mean(x) - use the method 'mean' on the data object 'x'



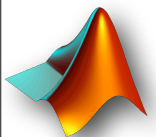
- A special method of a class which builds an instance of the class (builds an object)
 - normally the method has the same name as the class
- Examples:
 - `car('blue')` - builds a blue car
 - `animal('dog', 'brown')` - builds a brown dog which is a type of animal



construct



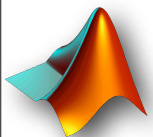
- A special method of a class which builds an instance of the class (builds an object)
 - normally the method has the same name as the class
- Examples:
 - `car('blue')` - builds a blue car
 - `animal('dog', 'brown')` - builds a brown dog which is a type of animal



property



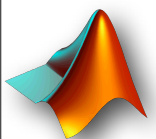
- A property is one aspect of a class (or object)
- Examples:
 - a 'car' might have properties:
 - make, color, top-speed, cost
 - an algorithm might have properties
 - input-type, different configuration parameters



property

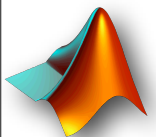
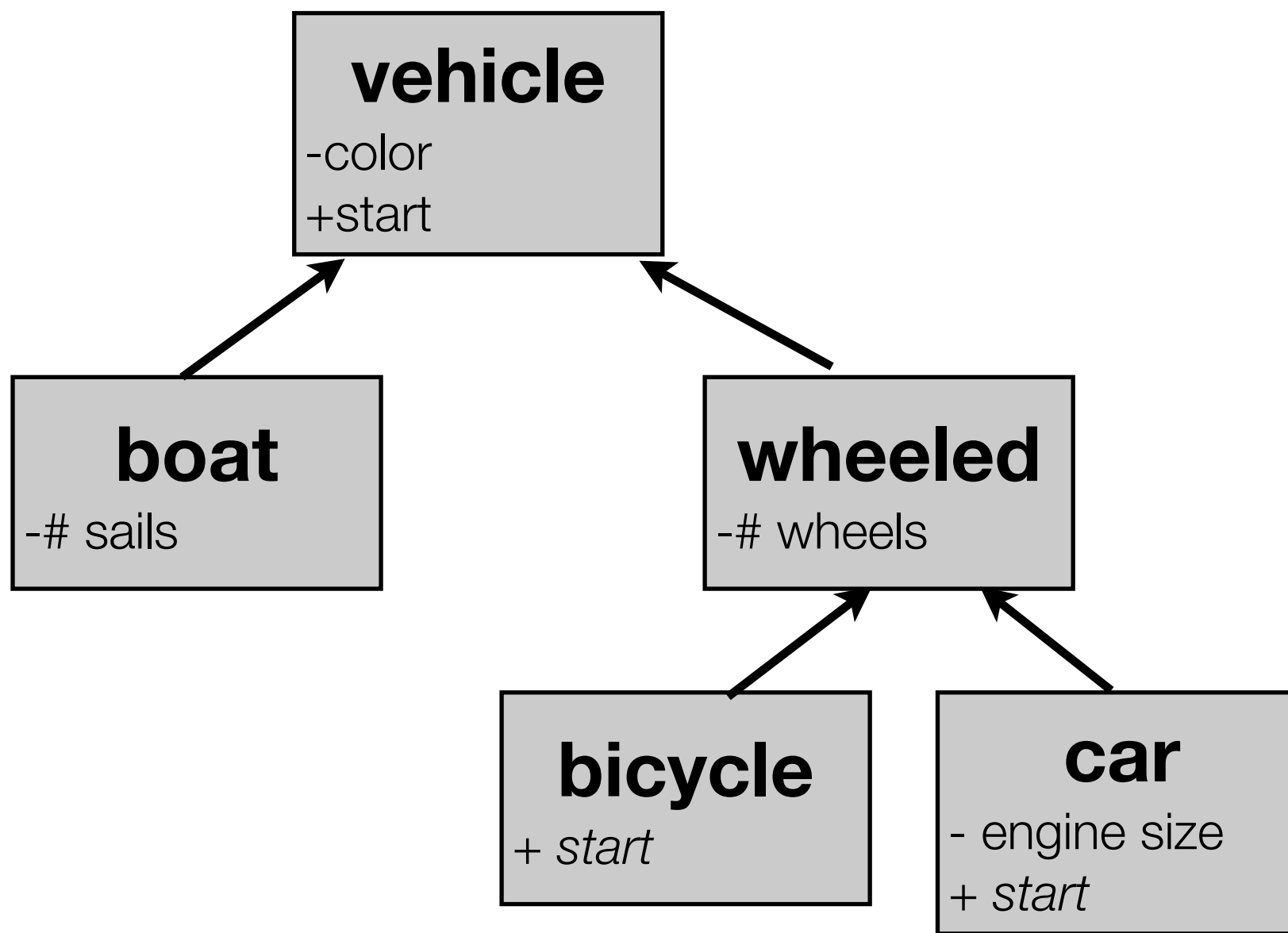


- A property is one aspect of a class (or object)
- Examples:
 - a 'car' might have properties:
 - make, color, top-speed, cost
 - an algorithm might have properties
 - input-type, different configuration parameters



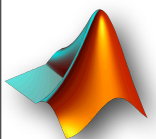
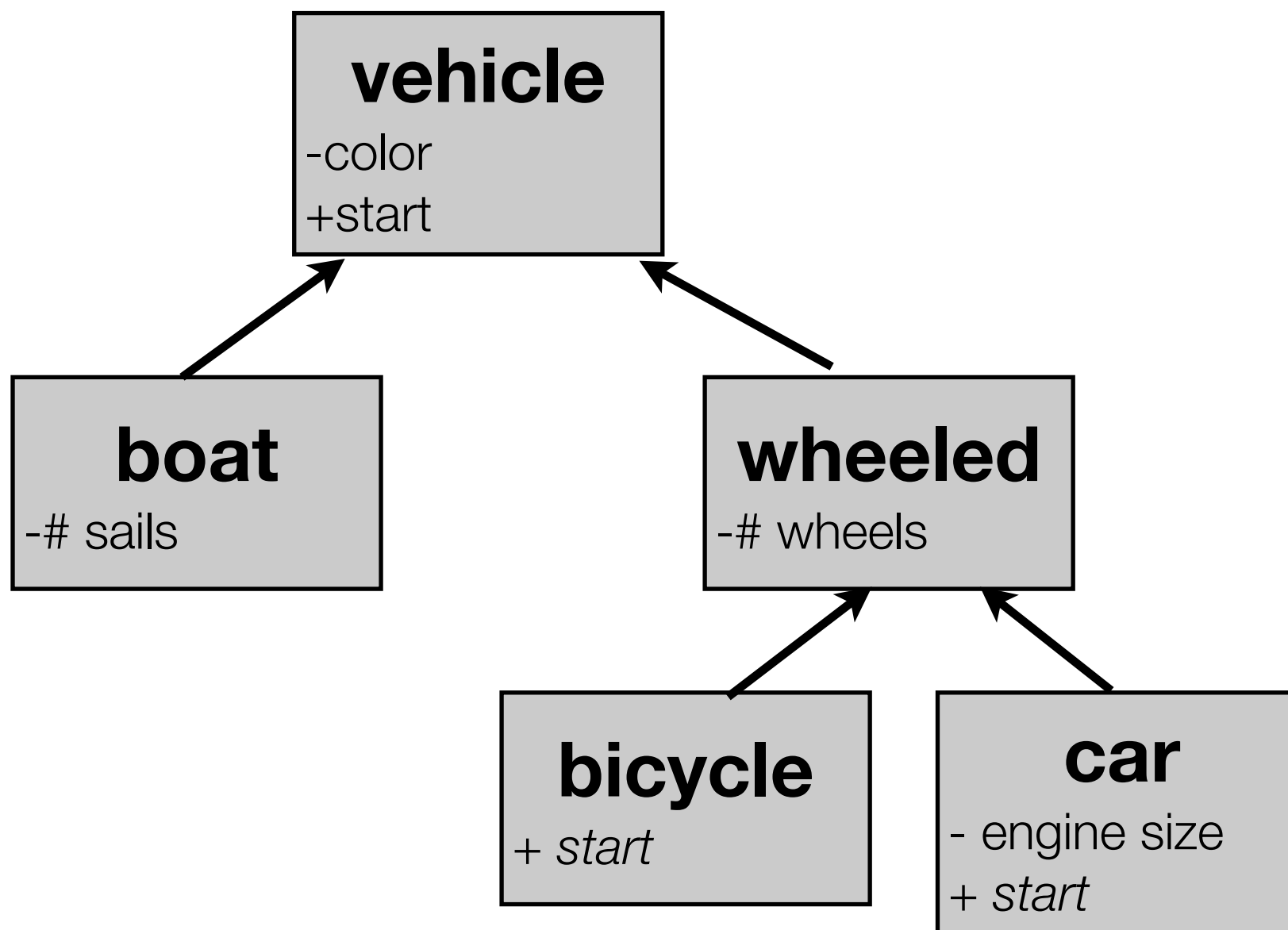
inheritance

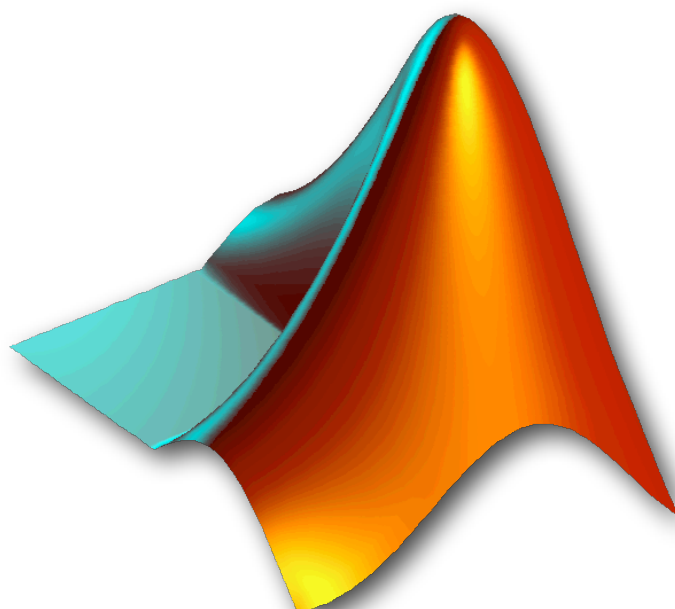
- classes can inherit behaviour (methods and properties) from other classes



inheritance

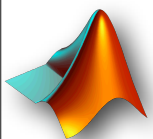
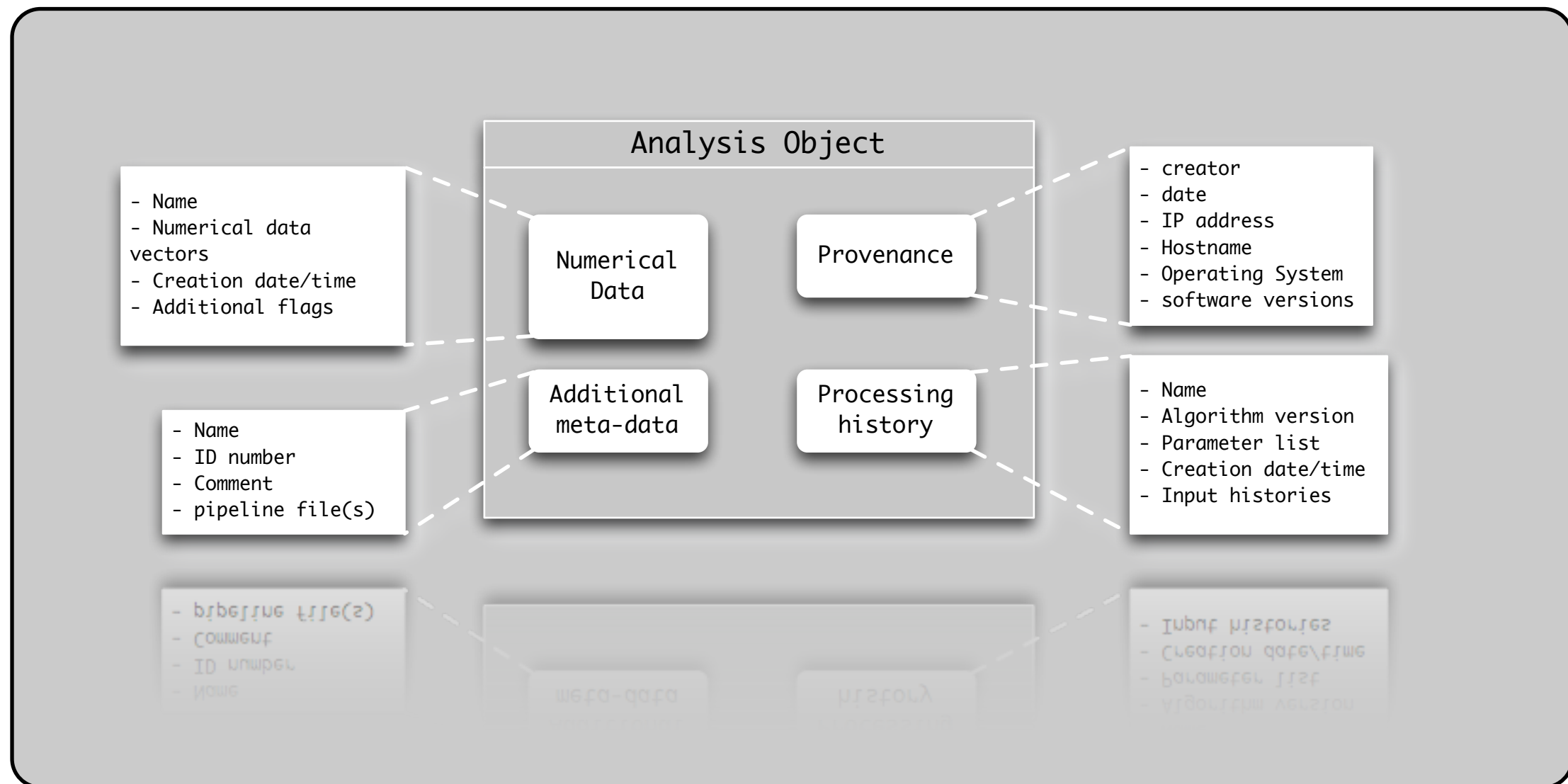
- classes can inherit behaviour (methods and properties) from other classes





Analysis Objects

- Aim to store data products

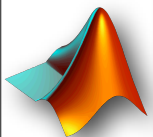


AO Methods

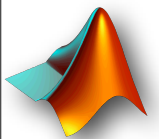


```
>> methods ao
```

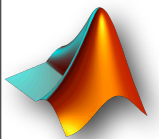
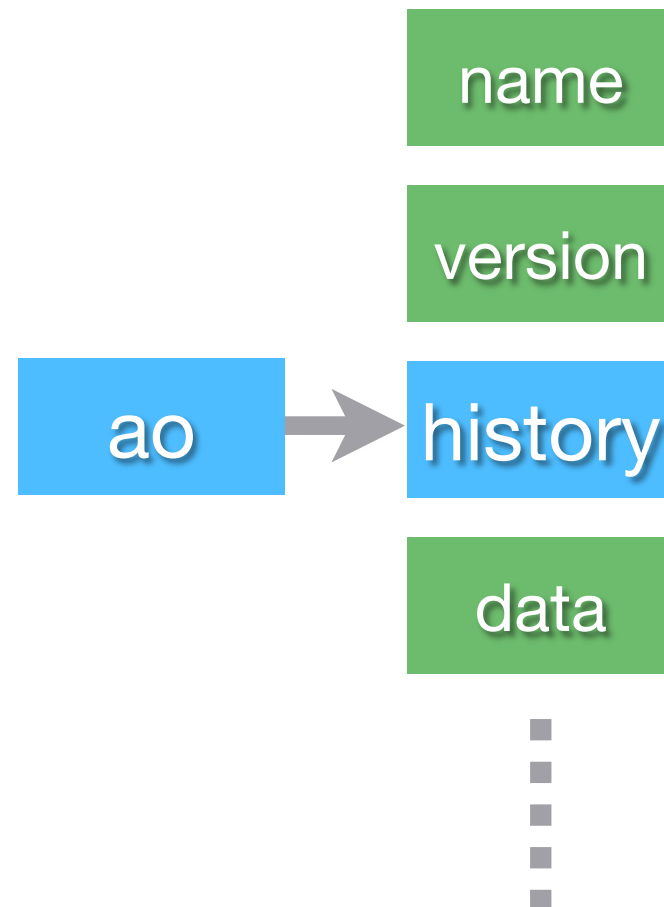
Contents	detrend	hist_gauss	mdc1_ifo2acc_inloop	rotate	std
abs	dft	hypot	mdc1_ifo2cont_utn	round	straightLineFit
acos	diag	ifft	mdc1_ifo2control	sDomainFit	string
angle	diff	imag	mdc1_x2acc	save	submit
ao	display	index	mean	scale	sum
asin	dopplercorr	integrate	median	scatterData	sumjoin
atan	downsample	interp	min	search	svd
atan2	dropduplicates	interpmissing	minus	select	svd_fit
bilinfit	dsmean	inv	mode	setDescription	t0
bin_data	dx	iplot	mpower	setDx	table
bsubmit	dy	iplotyy	mrdivide	setDy	tan
buildWhitener1D	eig	isprop	mtimes	setFs	tdfit
cat	eq	isvalid	ne	setMdlfile	tfe
char	eqmotion	join	noisegen1D	setName	timeaverage
cohere	evaluateModel	lcohere	noisegen2D	setPlotinfo	timedomainfit
complex	exp	lcpsd	norm	setProcinfo	times
compute	export	le	normdist	setT0	timeshift
confint	fft	len	nsecs	setUUID	transpose
conj	fftfilt	linSubtract	offset	setX	type
consolidate	filtSubtract	lincom	optSubtraction	setXY	uminus
conv	filter	linedetect	phase	setXunits	unwrap
convert	filtfilt	linfit	plot	setY	update
copy	find	lisovfit	plus	setYunits	upsample
corr	firwhiten	ln	polyfit	setZ	validate
cos	fixfs	log	polynomfit	sign	var
cov	fngen	log10	power	simplifyYunits	viewHistory
cpsd	fromProcinfo	lpsd	psd	sin	whiten1D
crbound	fs	lscov	psdconf	sineParams	whiten2D
created	gapfilling	lt	pwelch	smallvector_lincom	x
creator	gapfillingoptim	ltfe	quasiSweptSine	smallvectorfit	xcorr
csvexport	ge	ltp_ifo2acc	rdivide	smoother	xfit
ctranspose	get	max	real	sort	xunits
curvefit	getdof	mcmc	rebuild	spectrogram	y
delay	gnuplot	md5	removeVal	spikecleaning	yunits
delayEstimate	gt	mdc1_cont2act_utn	report	split	zDomainFit
demux	heterodyne	mdc1_ifo2acc_fd	resample	spsd	zeropad
det	hist	mdc1_ifo2acc_fd_utn	rms	sqrt	



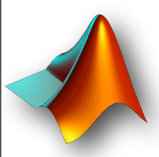
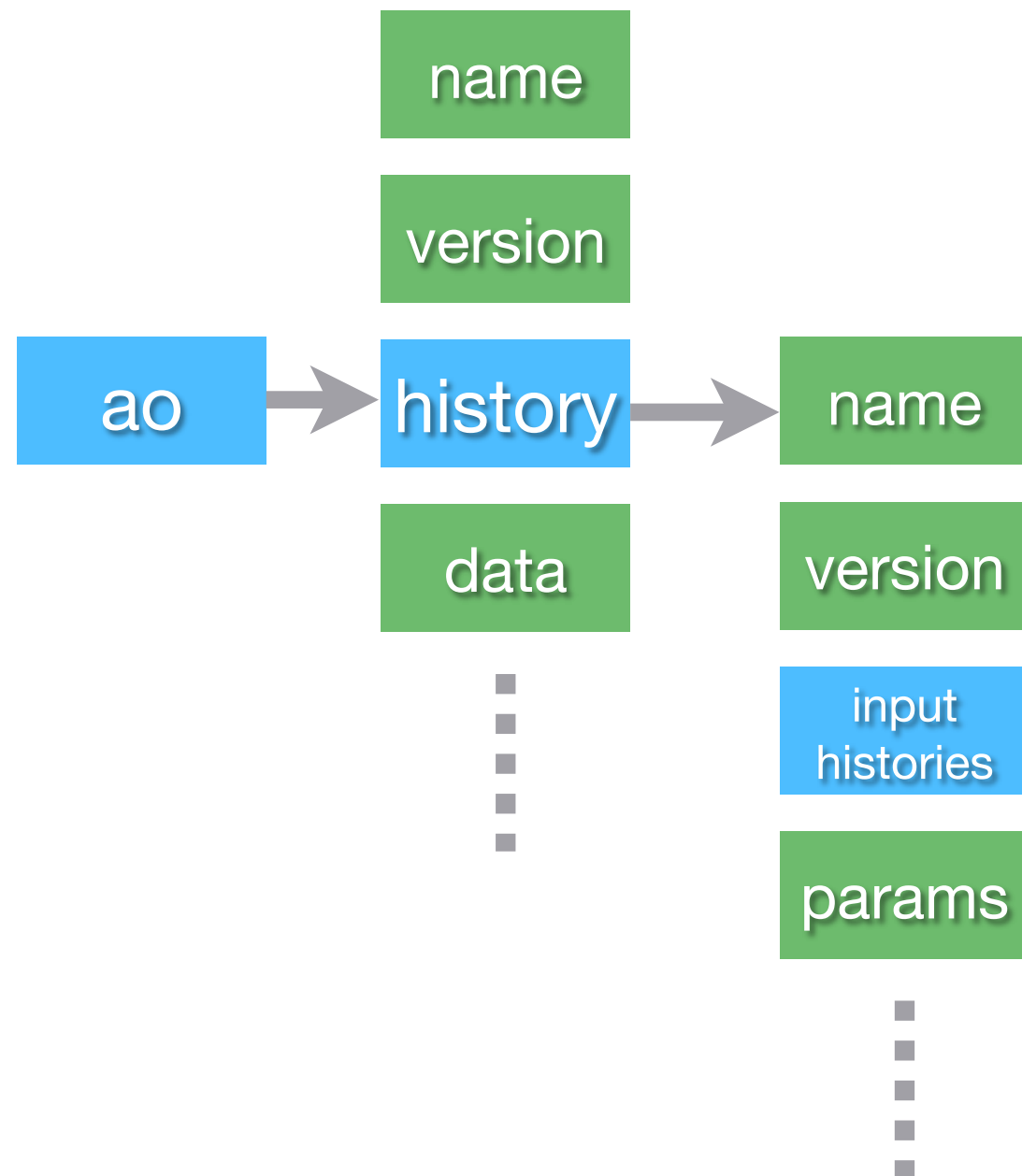
Tracking history...



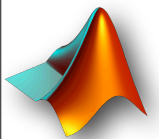
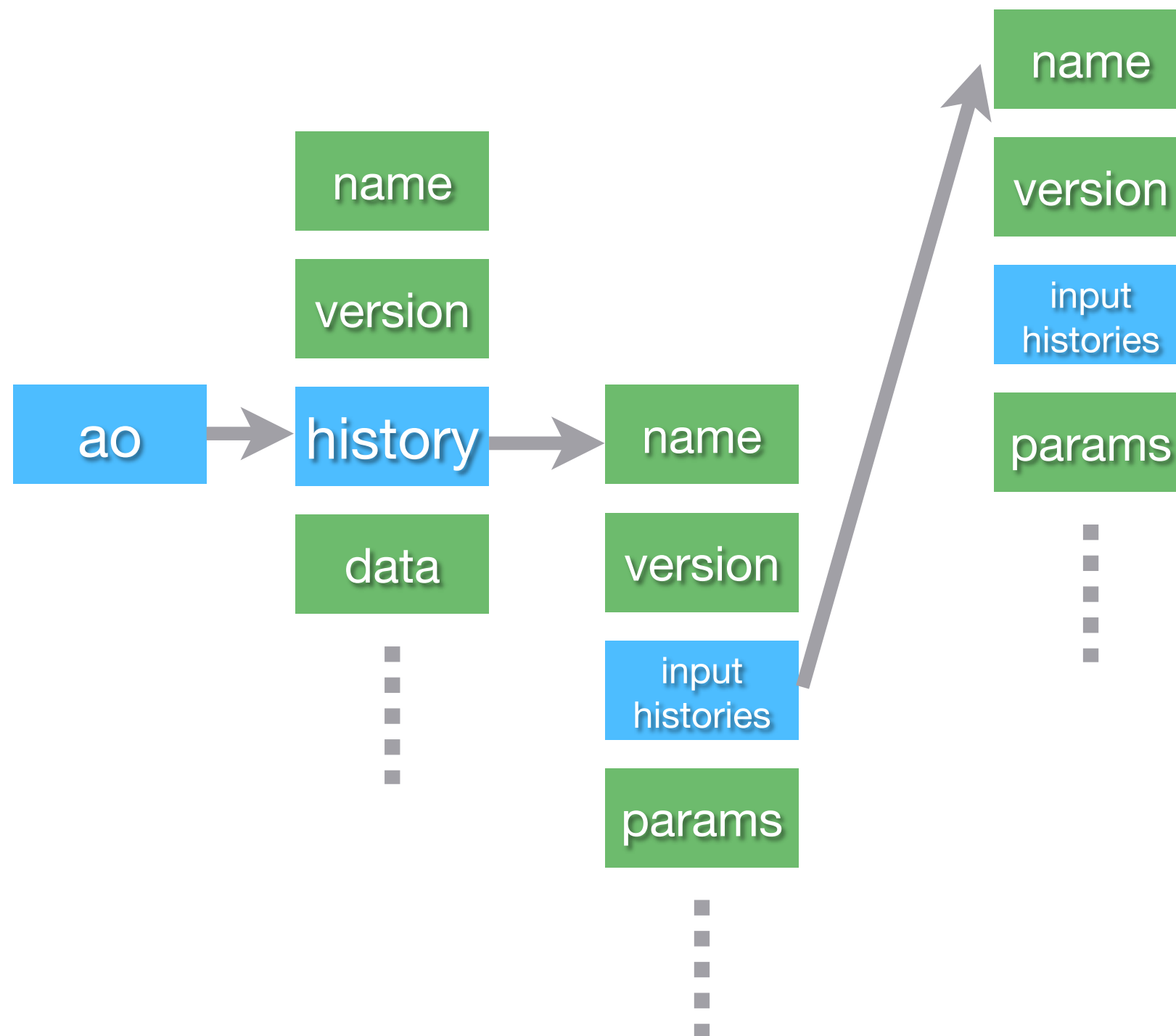
Tracking history...



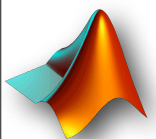
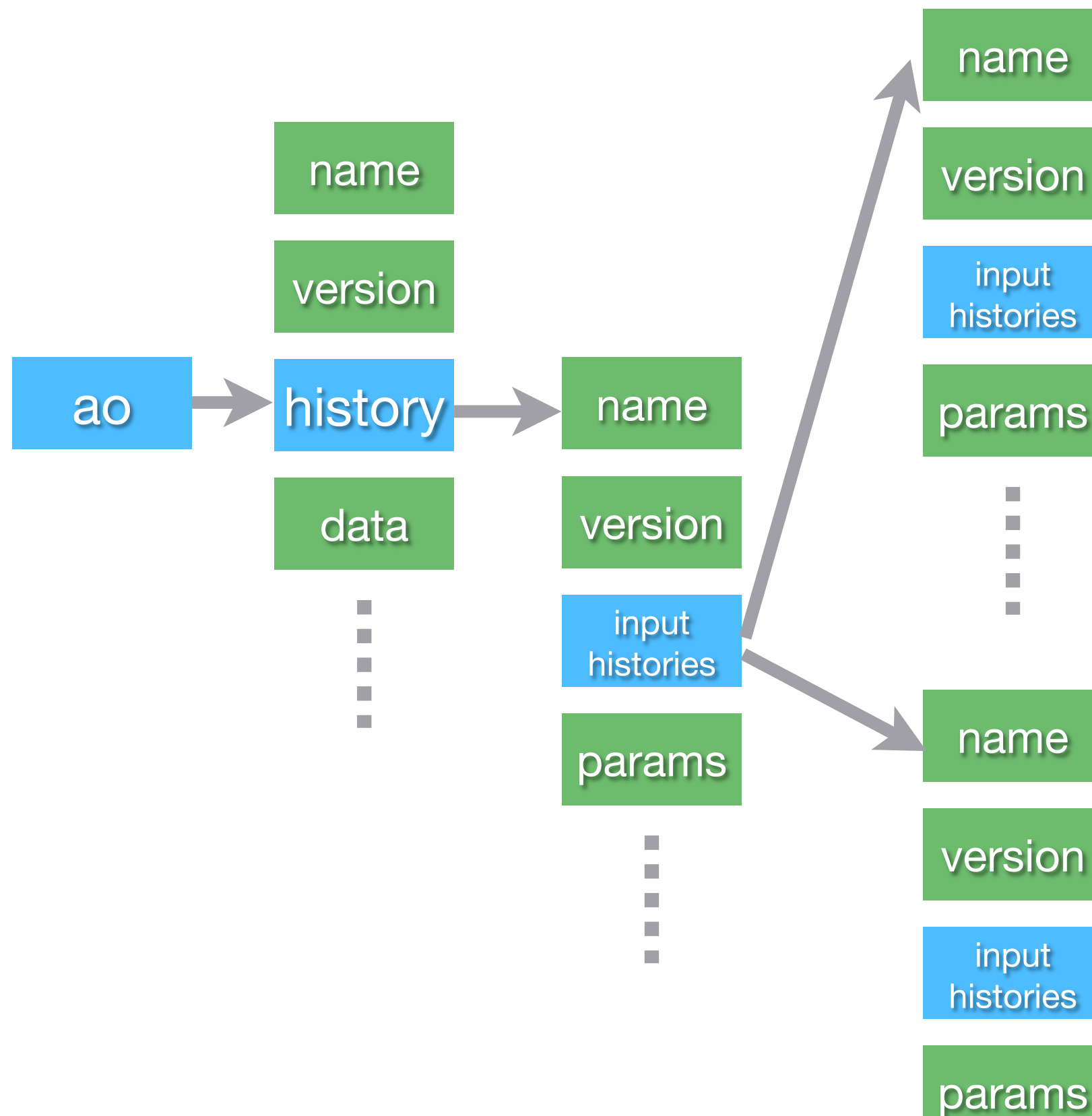
Tracking history...



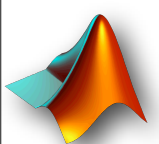
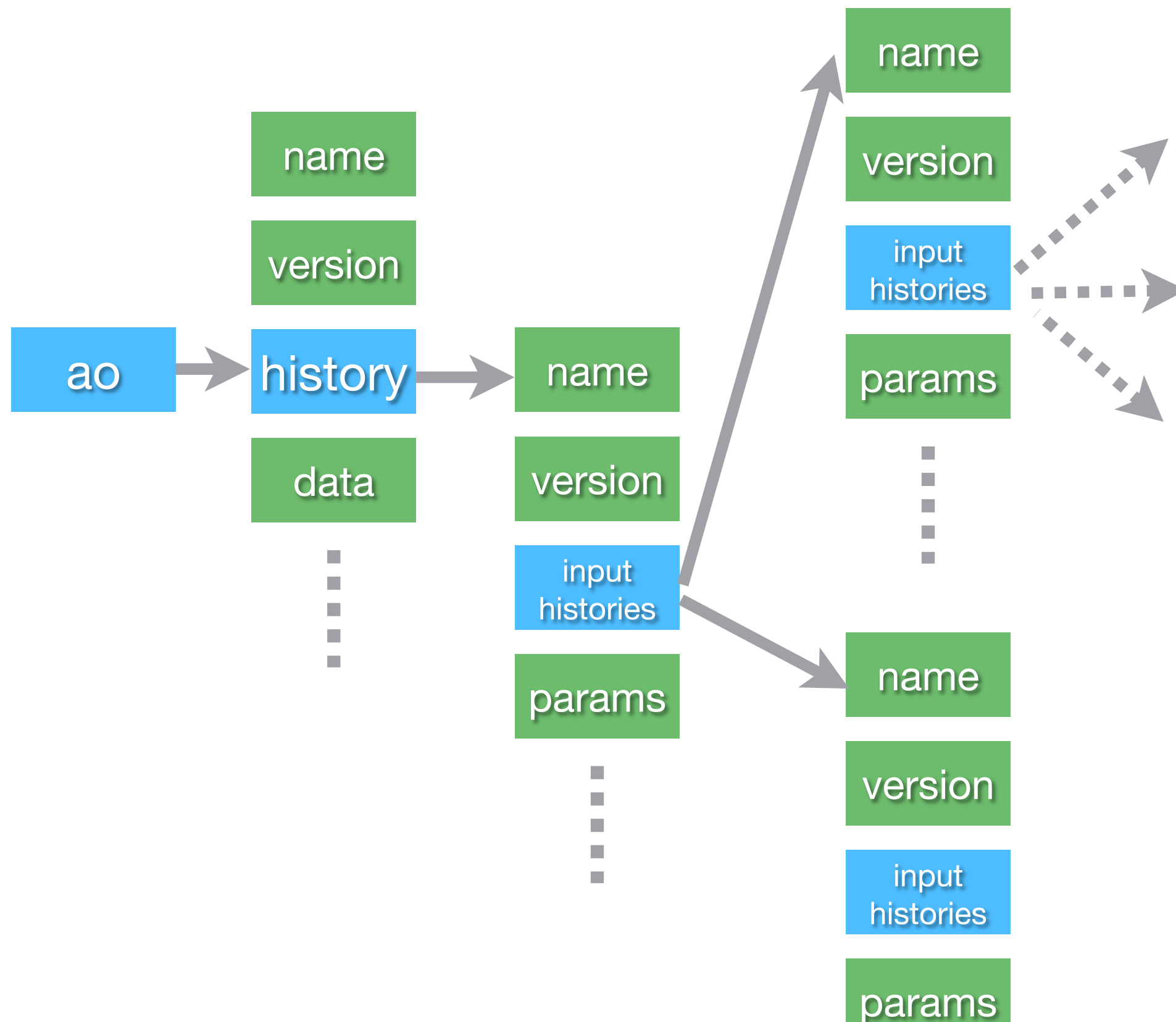
Tracking history...



Tracking history...



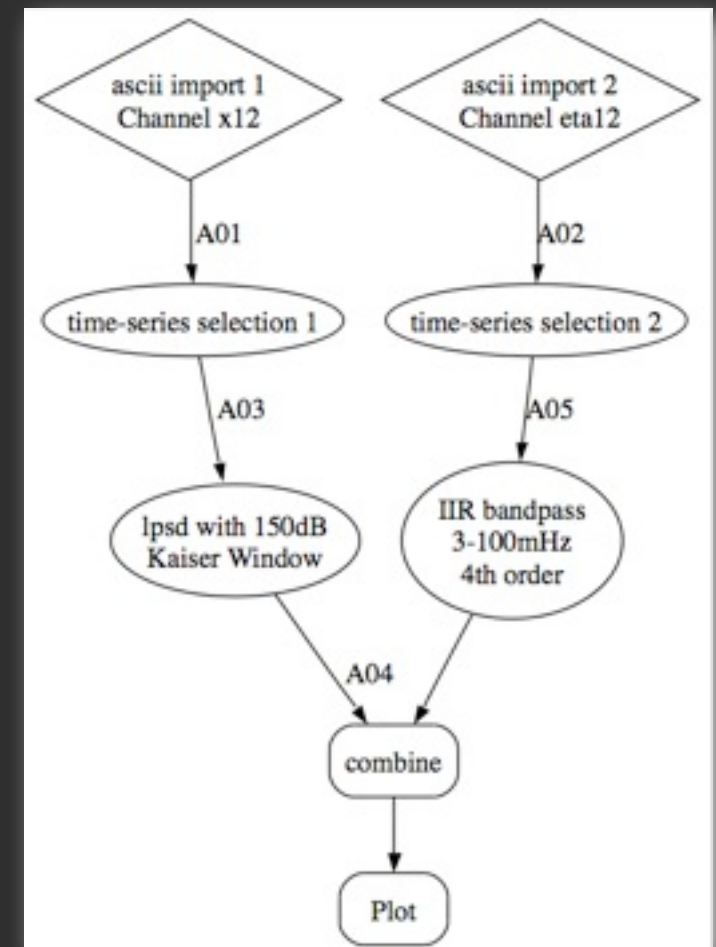
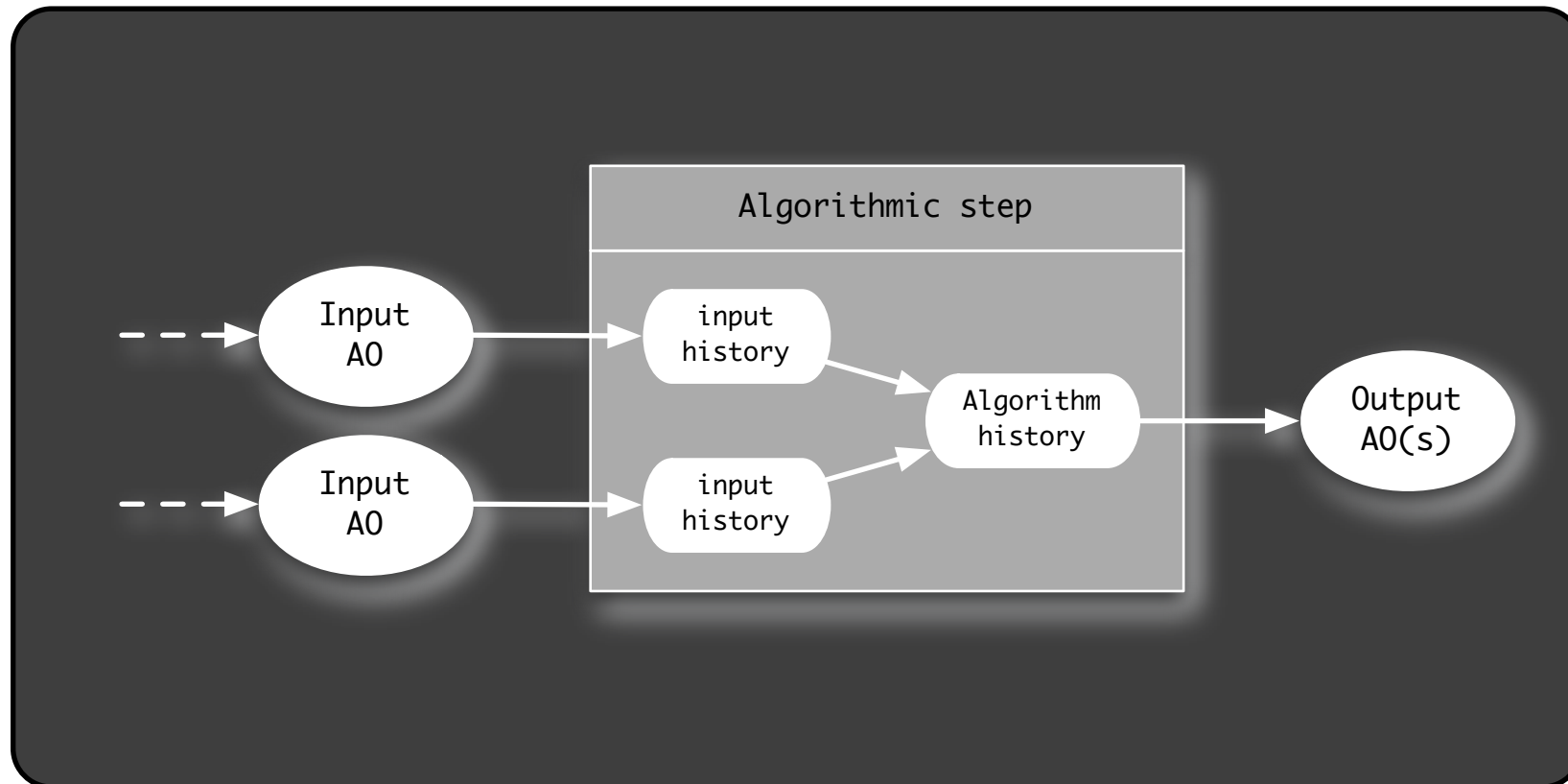
Tracking history...



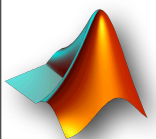
Smart algorithms



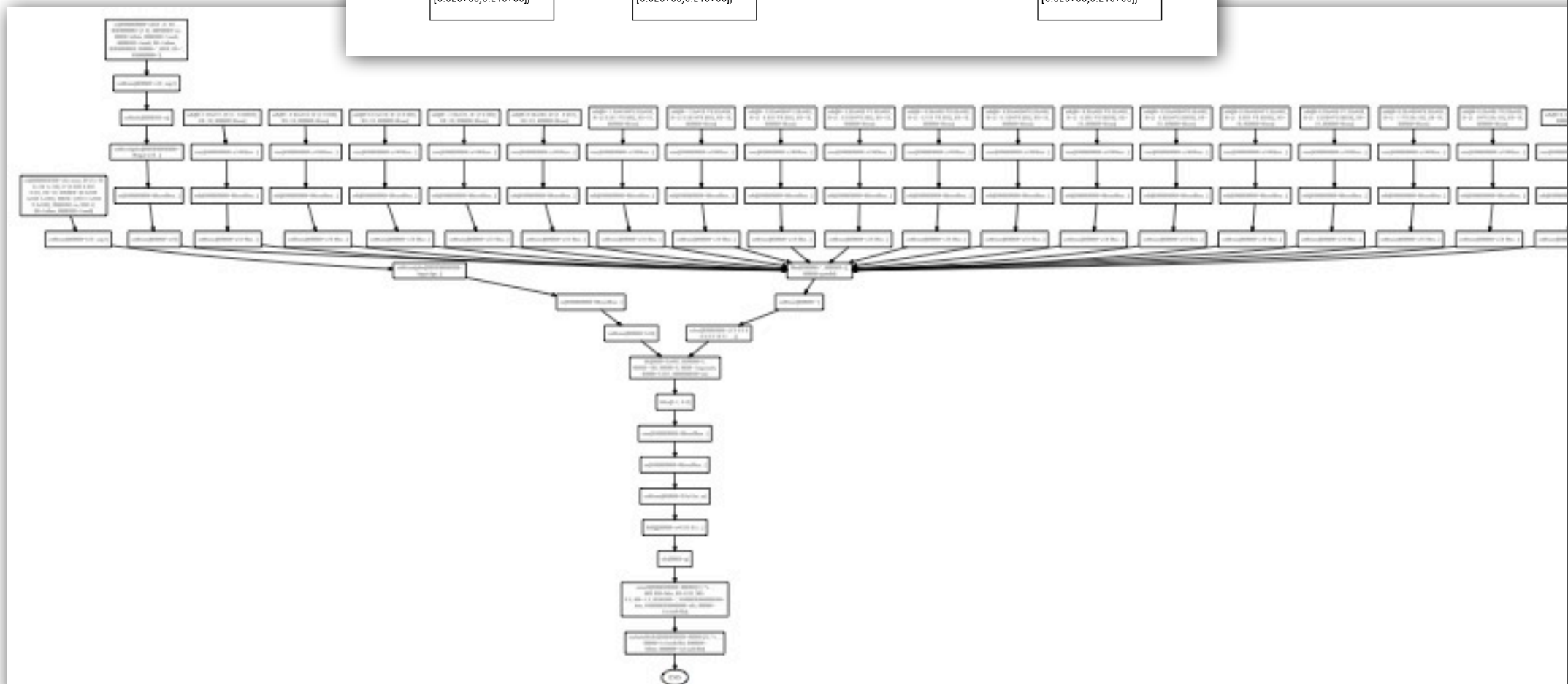
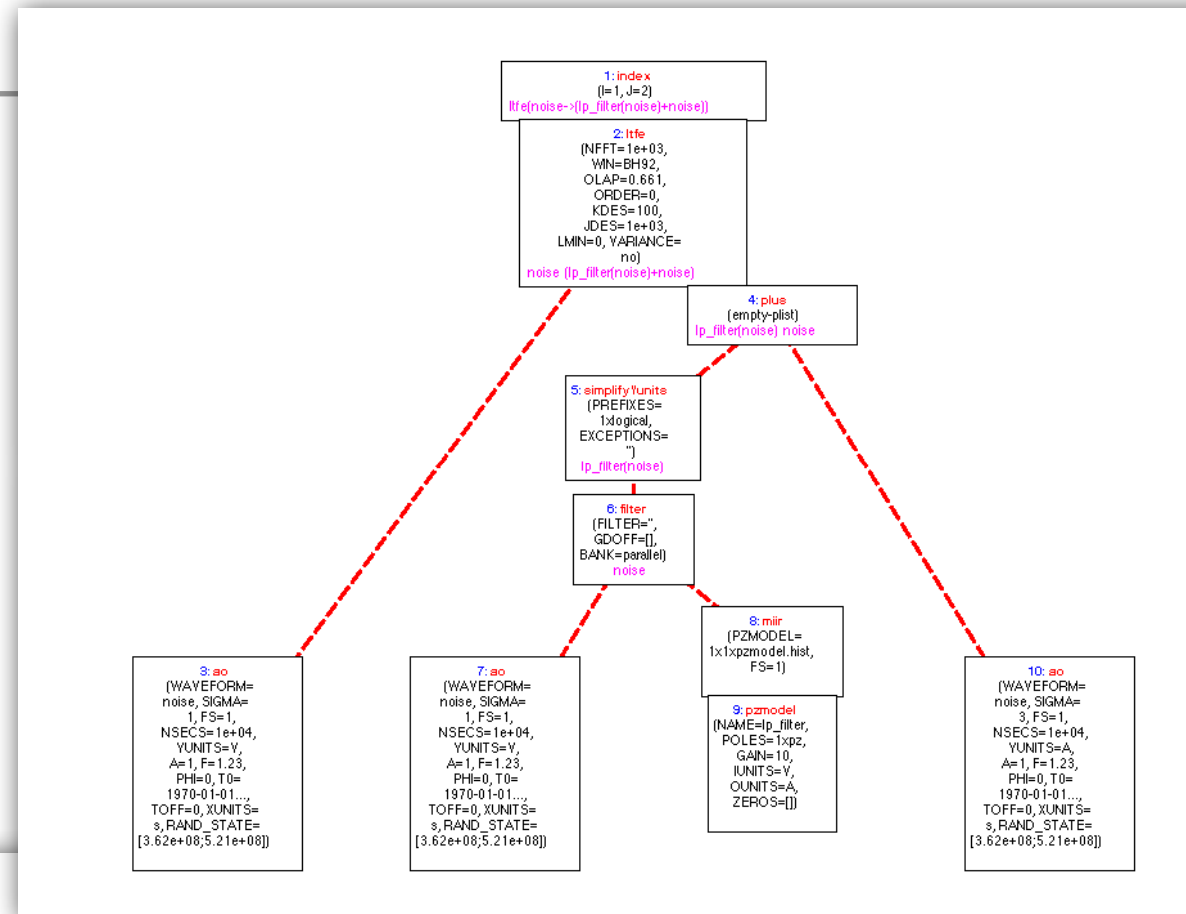
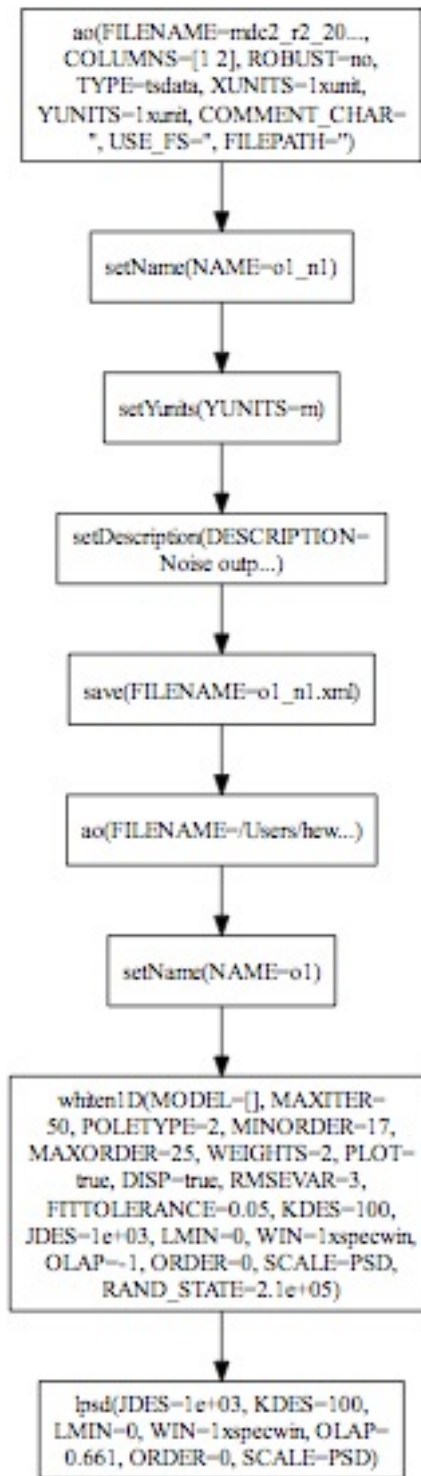
Intelligent algorithms



traceable results



Viewing history



Reliving history

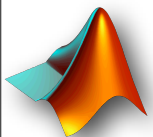


```
obj.type(<file>)
```

*output commands needed
to rebuild this object*

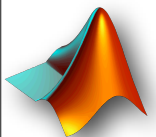
```
robject = obj.rebuild
```

rebuild this object



Objects, Objects, Everywhere...

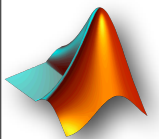
- The LTPDA Toolbox is fully object-oriented
- The user deals with:
 - 'LTPDA user objects'
 - MATLAB primitives (strings, doubles, logicals, etc)
- We have 13 LTPDA User Classes
 - 'ao', 'collection', 'filterbank', 'matrix', 'mfir', 'miir', 'parfrac', 'pest', 'pzmodel', 'rational', 'smodel', 'ssm', 'timespan'
- Two 'helper' classes
 - 'plist', 'time'



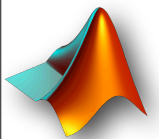
Class-diagram



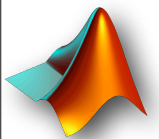
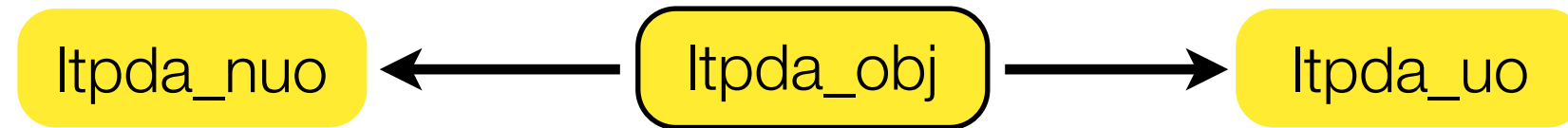
ltpda_obj



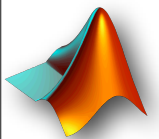
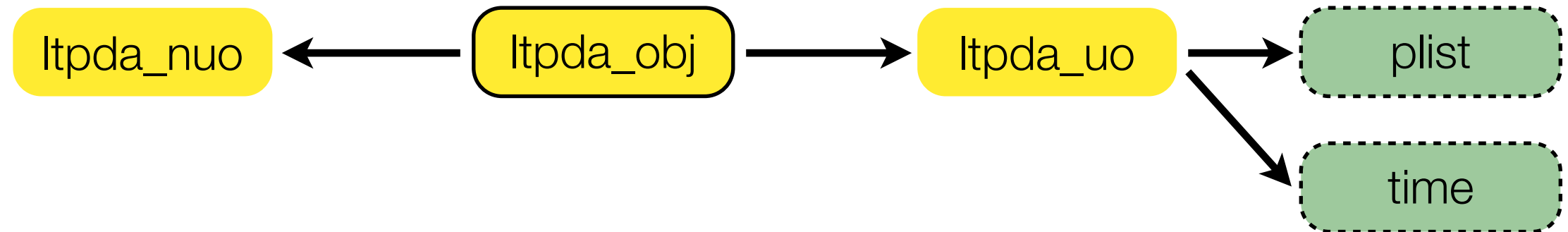
Class-diagram



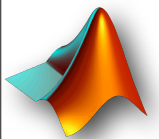
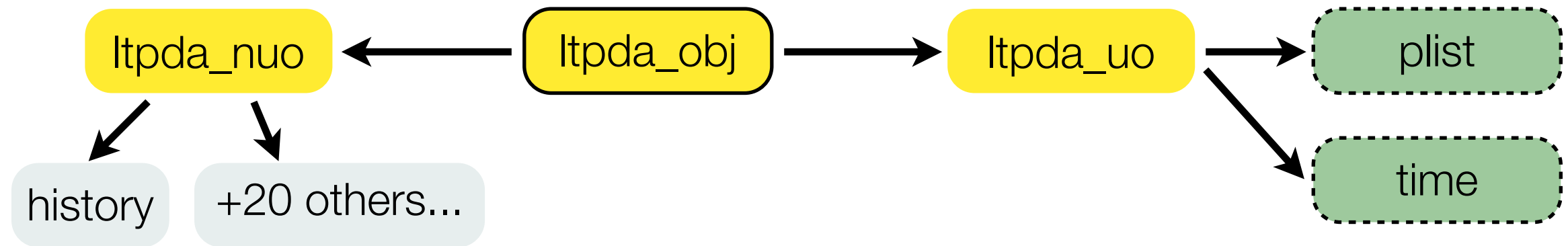
Class-diagram



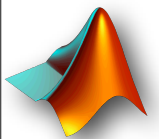
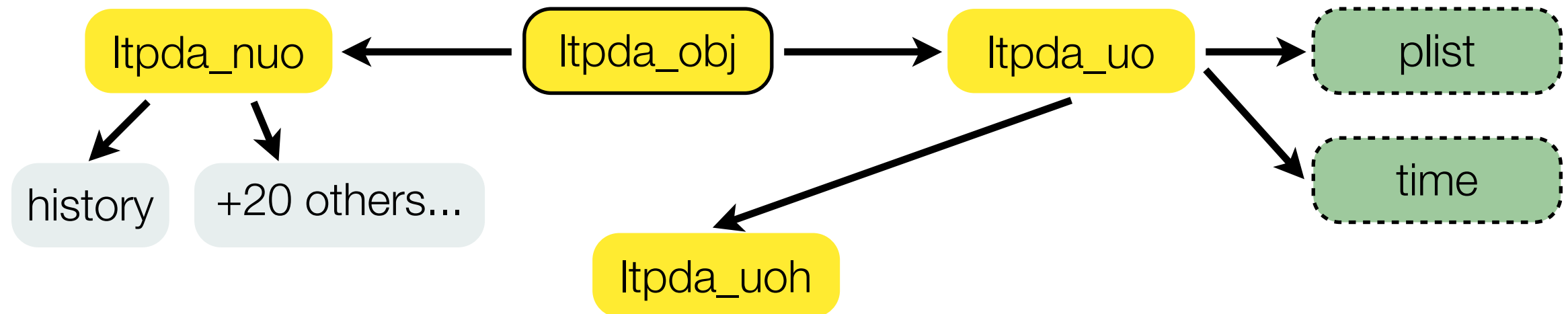
Class-diagram



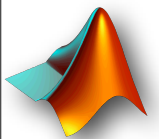
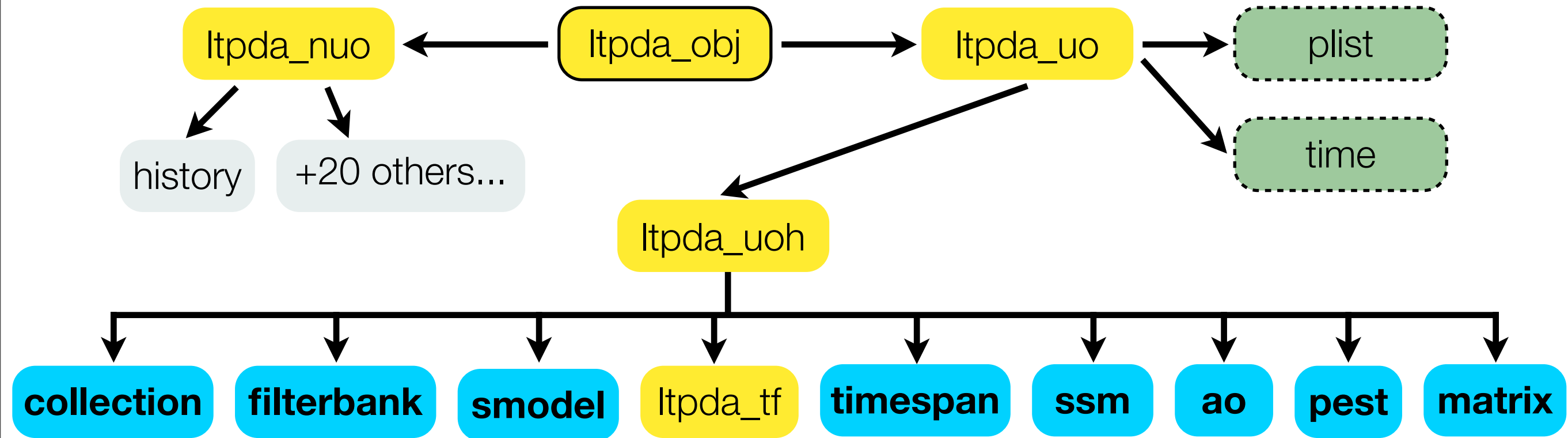
Class-diagram



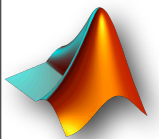
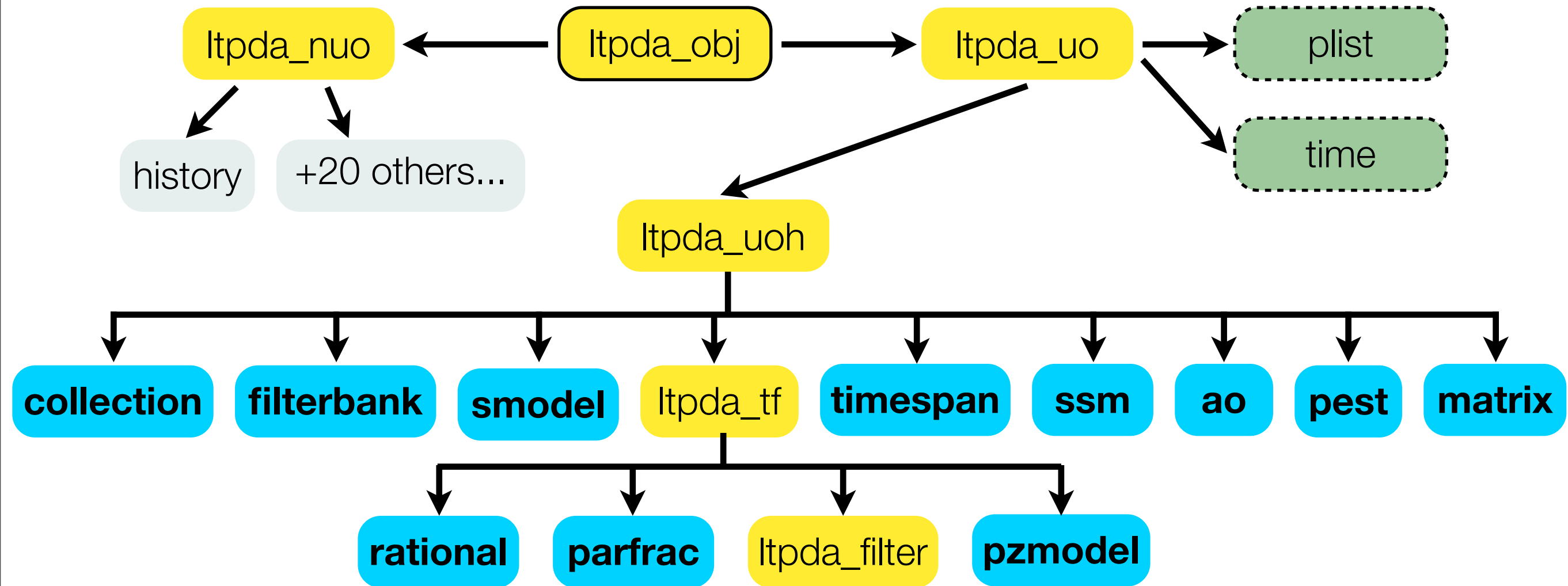
Class-diagram



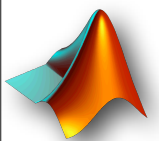
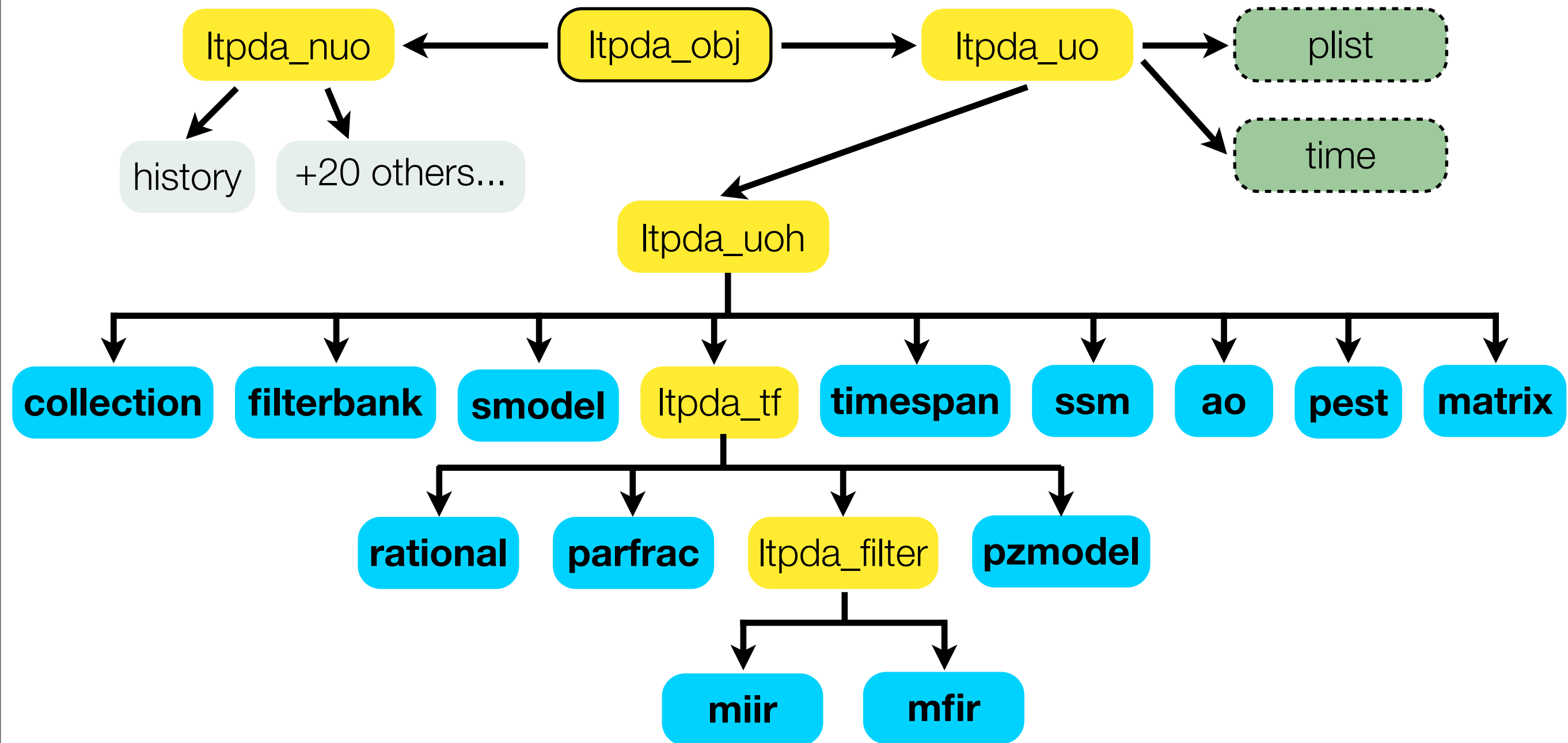
Class-diagram



Class-diagram



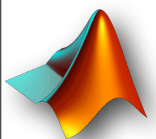
Class-diagram



Parameter lists

- Essentially all methods and constructors are *configured* by parameter lists (`pLists`)
- A parameter list has:
 - a list of parameters
 - a name (inherited)
 - a description (inherited)
 - a UUID* (inherited)
- Each parameter has a 'key' and a 'value'
 - key == string
 - value == MATLAB primitive or LTPDA object

*<http://en.wikipedia.org/wiki/UUID>



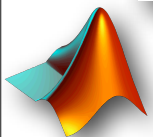
Examples:

```
>> pl = plist
----- plist 01 -----
n params: 0
description:
UUID:          9d9ef3a6-52f1-4fbf-a442-f83d9d7b6b88
-----
```

Empty Parameter List

```
>> pl = plist('a', 1, 'b', 'two')
----- plist 01 -----
n params: 2
---- param 1 ----
key:  A
val:  1
-----
---- param 2 ----
key:  B
val:  'two'
-----
description:
UUID:          d4b59063-292c-4254-b562-0fef3fb11c17
-----
```

Parameter list with
two parameters, 'a'
and 'b'



Building objects

- Objects are built using class constructors:
 - `object = <class_name>(<arguments>)`

- Examples:

```
>> a = ao
```

 empty analysis object

```
>> a = ao(1)
```

 analysis object with a single data value

```
>> a = ao(plist('vals', 1))
```

 as above, using a plist

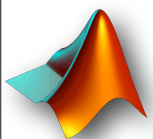
```
>> s = smodel('a*x+b')
```

 symbolic model of a straight-line

```
>> s = mfir('filter.xml')
```

 build an FIR filter by loading it from a file

```
>> pl = plist('filename', 'filter.xml')  
>> s = mfir(pl)
```

 as above, using a plist

Getting help



- How do I know which parameters to put in my plist?

```
>> help mfir
```

```
MFIR FIR filter object class constructor.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
DESCRIPTION: MFIR FIR filter object class constructor.  
             Create a mfir object.
```

```
CONSTRUCTORS:
```

```
    f = mfir()          - creates an empty mfir object.
```

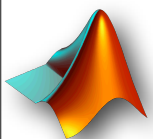
```
    :  
    :
```

```
Parameter Sets
```

```
VERSION:      $Id: mfir.m,v 1.103 2010/05/05 09:30:07 ingo Exp $
```

```
SEE ALSO:     miir, ltpda_filter, ltpda_uoh, ltpda_uo, ltpda_obj, plist
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



Getting help

- How do I know which parameters to put in my plist?

```
>> help mfir
```

```
MFIR FIR filter object class constructor.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
DESCRIPTION: MFIR FIR filter object class constructor.  
             Create a mfir object.
```

```
CONSTRUCTORS:
```

```
    f = mfir()           - creates an empty mfir object.
```

```
    :
```

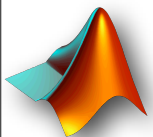
```
    Parameter Sets
```

 *click here*

```
VERSION:      $Id: mfir.m,v 1.103 2010/05/05 09:30:07 ingo Exp $
```

```
SEE ALSO:     miir, ltpda_filter, ltpda_uoh, ltpda_uo, ltpda_obj, plist
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```





Method Report for mfir/mfir

Some information of the method mfir/mfir are listed below:

Class name	mfir
Method name	mfir
Category	Constructor
CVS Version	\$Id: mfir.m,v 1.103 2010/05/05 09:30:07 ingo Exp \$
Min input args	0
Max input args	-1
Min output args	1
Max output args	1

Sets for this method ...

- [Default](#)
- [From MAT File](#)
- [From XML File](#)
- [From Repository](#)
- [From Built-in Model](#)
- [From Standard Type](#)
- [From Pzmodel](#)
- [From A](#)
- [From AO](#)

Default

Key	Default Value	Options	Description
NAME	'None'	<i>none</i>	The name of the constructed FIR filter.
DESCRIPTION	"	<i>none</i>	The description of the constructed FIR filter.

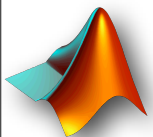
Getting more help

- LTPDA Toolbox has a decent amount of documentation:
 - `>> doc ltpda`
- Which methods are available?
 - `>> methods <class_name>`
 - example:

```
>> methods mfir
```

```
Methods for class mfir:
```

Contents	display	mfir	setA	setPlotinfo	update
bsubmit	eq	ne	setDescription	setProcinfo	viewHistory
char	get	rebuild	setHistout	setUUID	
copy	impresp	redesign	setIunits	simplifyUnits	
created	index	report	setMdlfile	string	
creator	isprop	resp	setName	submit	
csvexport	isvalid	save	setOunits	type	



Setting object properties

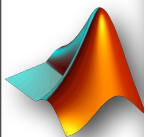
- Properties of an object can be set using 'setter' methods, or during construction:

```
>> a = ao(plist('name', 'bob'))
----- ao 01: bob -----

name: bob
data: None
hist: ao / ao / $Id: ao.m,v 1.315 2010/06/25 13:55:38 ingo Exp $
mdlfile: empty
description:
  UUID: 500458d6-2a4d-42a9-8d25-3fe5ebd1548f
-----
```

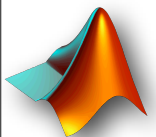
```
>> a = ao;
>> a.setName('bob')
----- ao 01: bob -----

name: bob
data: None
hist: ltpda_uoh / setName / $Id: setName.m,v 1.12 2010/06/07 16:35:26
ingo Exp $
mdlfile: empty
description:
  UUID: 7172543d-e69d-4fcf-abdd-b111b9c16434
-----
```



Modifying or copying...

- Many methods can be used to modify existing objects; some methods create new objects
- Modifying:
 - `>> a.setName('bob')`
 - the object 'a' will be modified and its name changed
- Copying:
 - `>> b = a.setName('bob')`
 - object 'a' will be copied. The copy will get the name 'bob' and 'a' will be left intact
- Some methods can not be used as modifiers

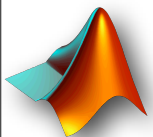


Help!



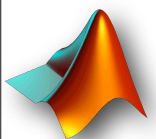
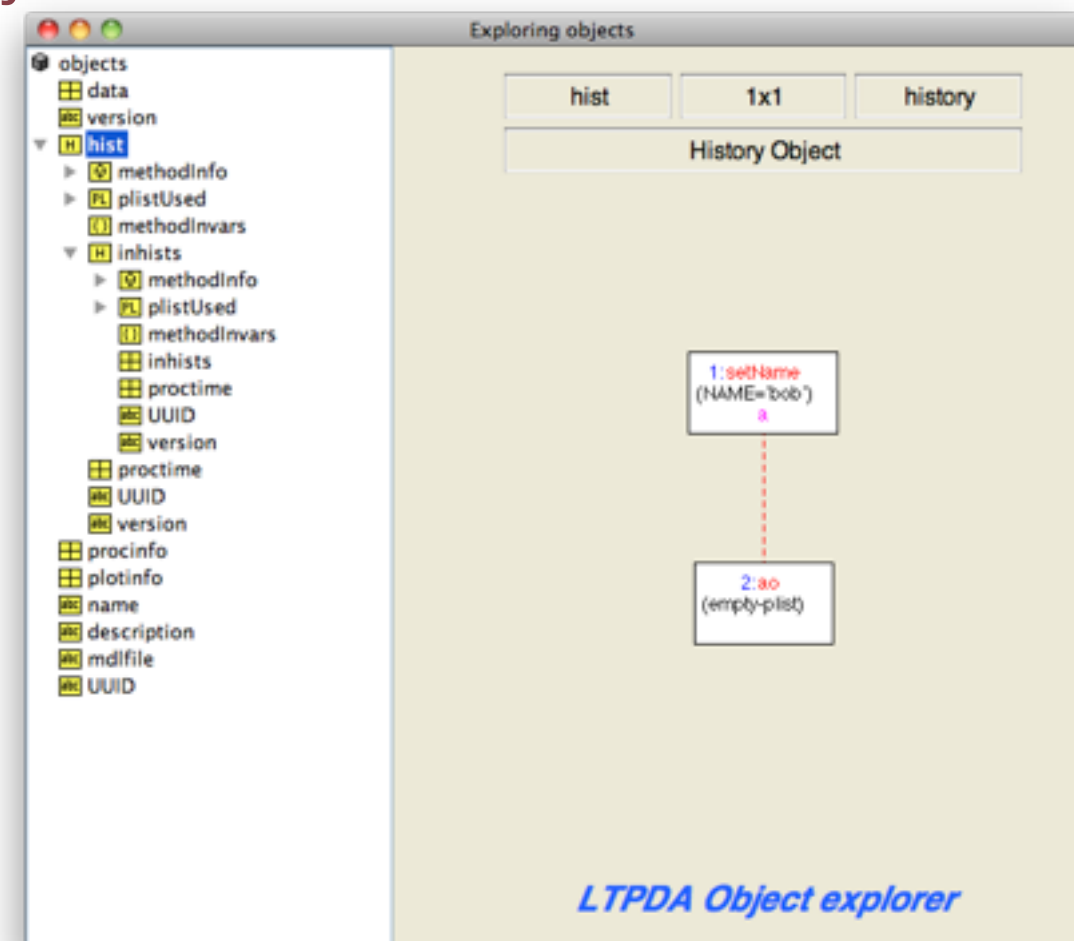
- Which properties does an object have?
- `>> properties <class_name>`
- `>> properties(object)`

Note: some properties are read-only. You need to check for the existence of a setter method like 'setName'. These methods take care of the history for you!



Viewing history

- We have two static ‘viewers’:
 - using graphviz (recall the introduction)
 - outputs vector graphics so can be used for huge history trees
 - using matlab
 - suitable for small history trees only
- Use graphical explorer
 - `>> ltpda_explorer(obj)`
- Look at the commands:
 - `>> type(obj)`



Viewing history

- We have two static ‘viewers’:
 - using graphviz (recall the introduction)
 - outputs vector graphics so can be used for huge history trees
 - using matlab
 - suitable for small history trees only

• Use

• >

• Loc

• >

```
>> type(a)
```

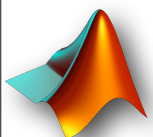
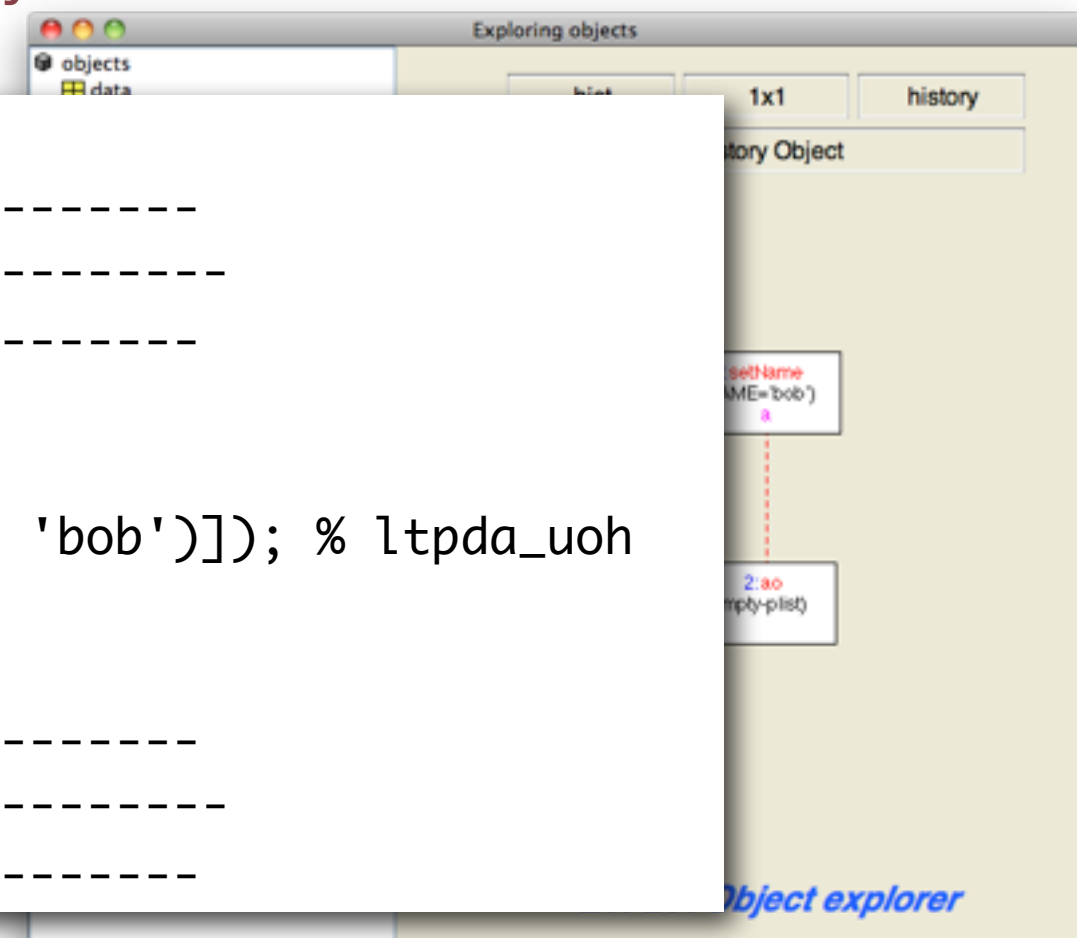
```
-----
-----bob-----
-----
```

```
a2812886 = ao([plist()]); % ao
```

```
a2818295 = setName(a2812886, [plist('NAME', 'bob')]); % ltpda_uoh
```

```
a_out = a2818295;
```

```
-----
-----bob-----
-----
```

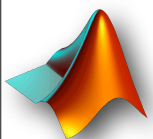


Build a time-series AO

- AOs can contain different types of data
 - Time-series data are stored in a `tsdata` object
 - In this case, the `ao.data` field will be a `tsdata` object
 - They also have properties:

tsdata	
t0	Absolute time-stamp of first sample
xunits	X-axis units
yunits	Y-axis units

- Constructors:
 - `a = ao(vector, sample_rate)`
 - `a = ao(plist('tsfcn', 't.^2 + t', 'fs', 10, 'nsecs', 1000))`
 - others: `>> help ao`, click 'Parameter Sets'



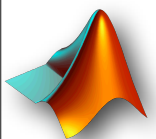
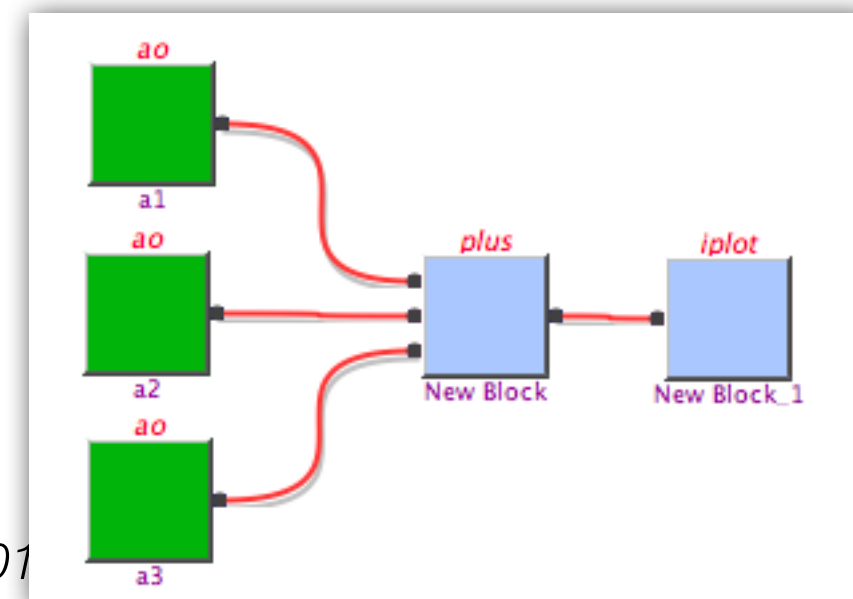
Basic Math

- You can operate on AOs using a large set of methods
 - In particular, many typical Math operations are available (overloaded)
 - Further details at: http://www.lisa.aei-hannover.de/ltpda/documents/files/operator_rules.pdf

```
a = ao(1);
b = ao(2);
c = a+b
```

```
a = ao(2);
a.^2
```

```
a = [ao(1) ao(2) ao(3)];
b = ao(4);
c = a + b
```

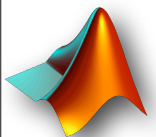


Saving and loading objects

- All LTPDA User Objects can be saved to (and loaded from), file in
 - XML format
 - binary MAT format

```
>> save(a, 'foo.xml')
>> save(a, 'foo.mat')
>> a.save(plist('filename', 'foo.xml'))
>> b = ao('foo.xml')
>> c = ao('foo.mat')
>> d = ao(plist('filename', 'foo.xml'))
```

```
<?xml version="1.0" encoding="utf-8"?>
<ltprda_object ltprda_version="2.0 (R2008b)">
  <object shape="1x1" type="ao">
    <property prop_name="data" shape="1x1" type="fsdata">
      <object shape="1x1" type="fsdata">
        <property prop_name="t0" shape="1x1" type="time">
          <object shape="1x1" type="time">
            <property prop_name="utc_epoch_milli" shape="1x1" type="double">1000</property>
            <property prop_name="timezone" shape="1x1" type="char">UTC</property>
            <property prop_name="timeformat" shape="1x23" type="char">%Y-%m-%d %H:%M:%S</property>
            <property prop_name="time_str" shape="0x0" type="char"></property>
            <property prop_name="version" shape="1x53" type="char">2.0 (R2008b)</property>
          </object>
        </property>
        <property prop_name="navs" shape="1x1" type="double">1</property>
        <property prop_name="fs" shape="1x1" type="double">1000</property>
        <property prop_name="enbw" shape="1x1" type="double">0.2</property>
        <property prop_name="version" shape="1x55" type="char">2.0 (R2008b)</property>
        <property prop_name="xunits" shape="1x1" type="unit">
          <object shape="1x1" type="unit">
            <property prop_name="strs" shape="1x1" type="cell">
              <property prop_name="unit" shape="1x1" type="char">m</property>
            </property>
          </object>
        </property>
      </object>
    </property>
  </object>
</ltprda_object>
```

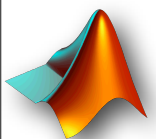


Reading existing data files

- You can construct AOs from existing ASCII (raw) data files

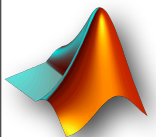
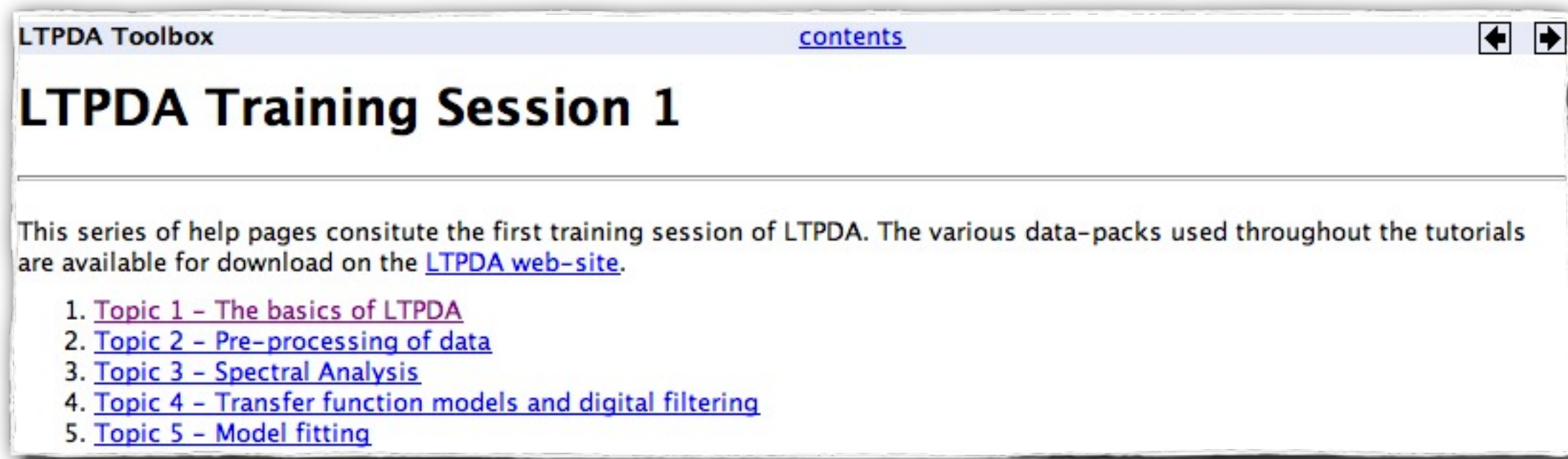
```
a = ao('topic1/simpleASCII.txt')
```

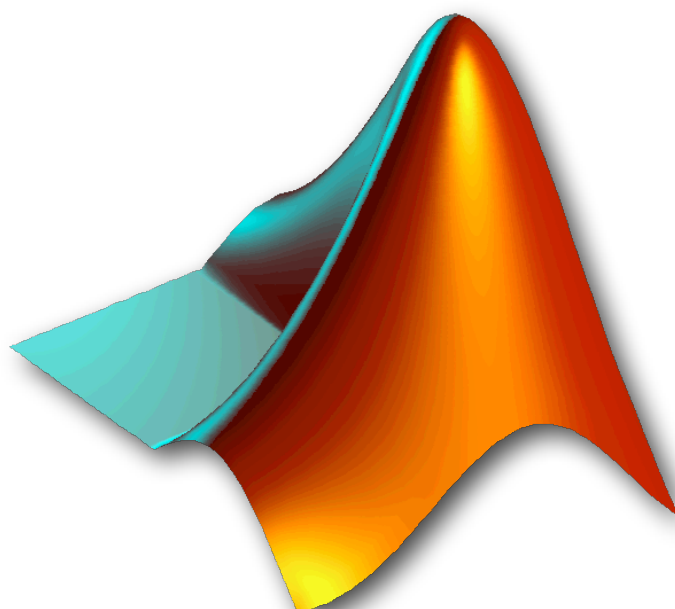
```
sigs = ao(plist('filename', 'topic1/multicolumnASCII.txt', ...  
               'columns', [1 3 1 5], ...  
               'name', {'sin2', 'sin4'}, ...  
               'yunits', {'m', 'm'}, ...  
               'description', {'sine wave at 2Hz', 'sine wave at 4Hz'}))
```



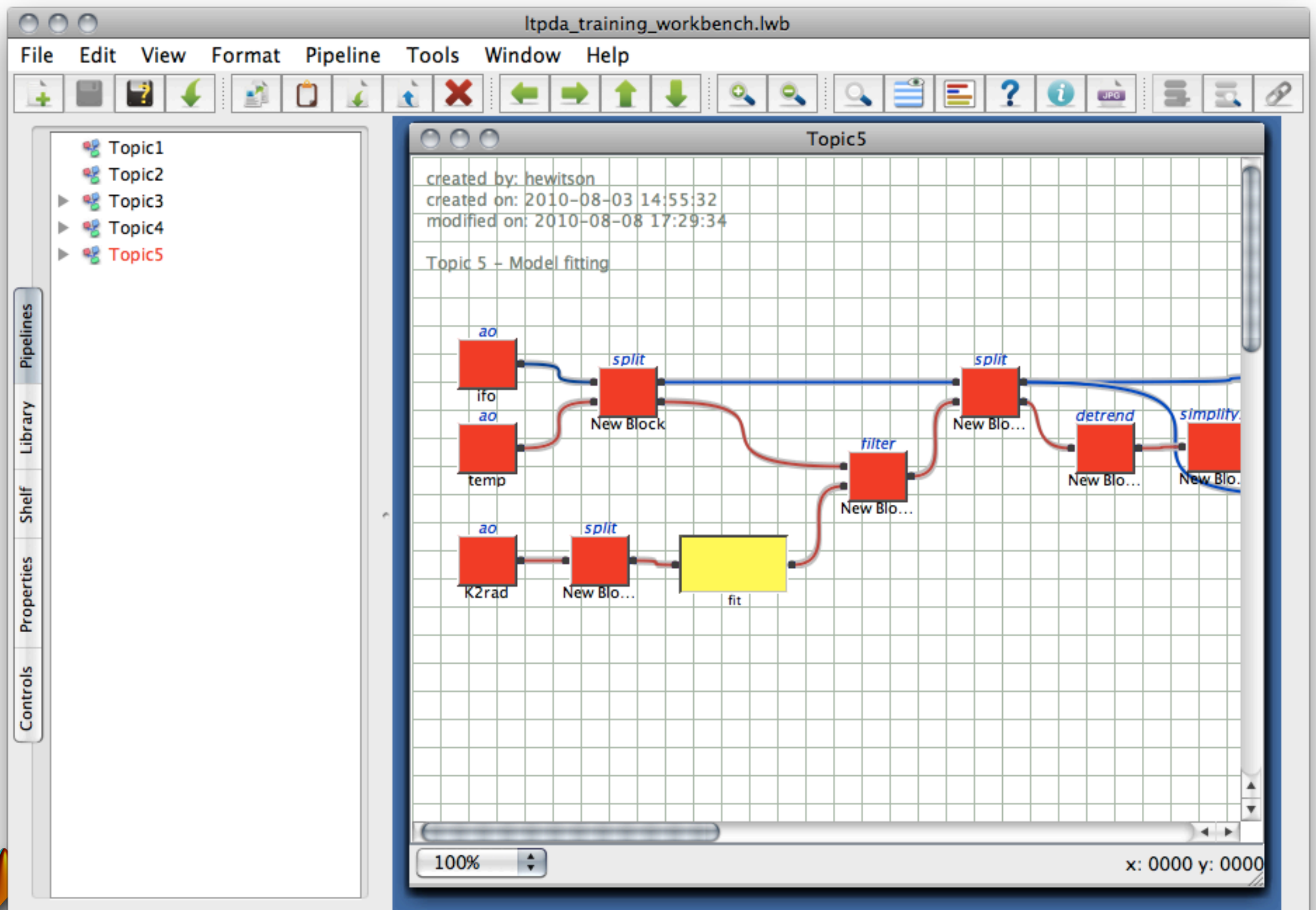
Hands on the keys...

- You can work through these concepts in the relevant section of the documentation
 - “LTPDA Training Session 1”

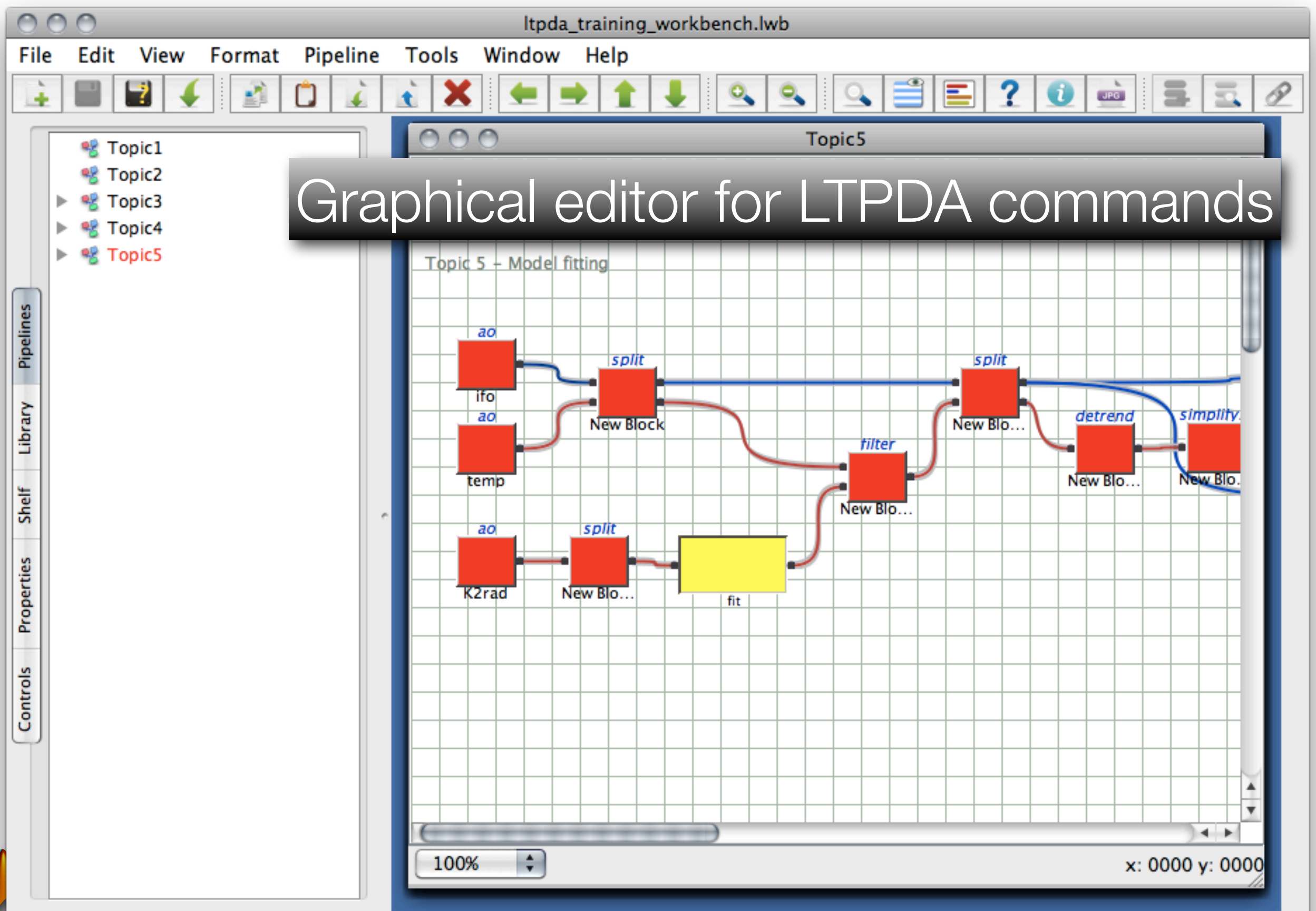




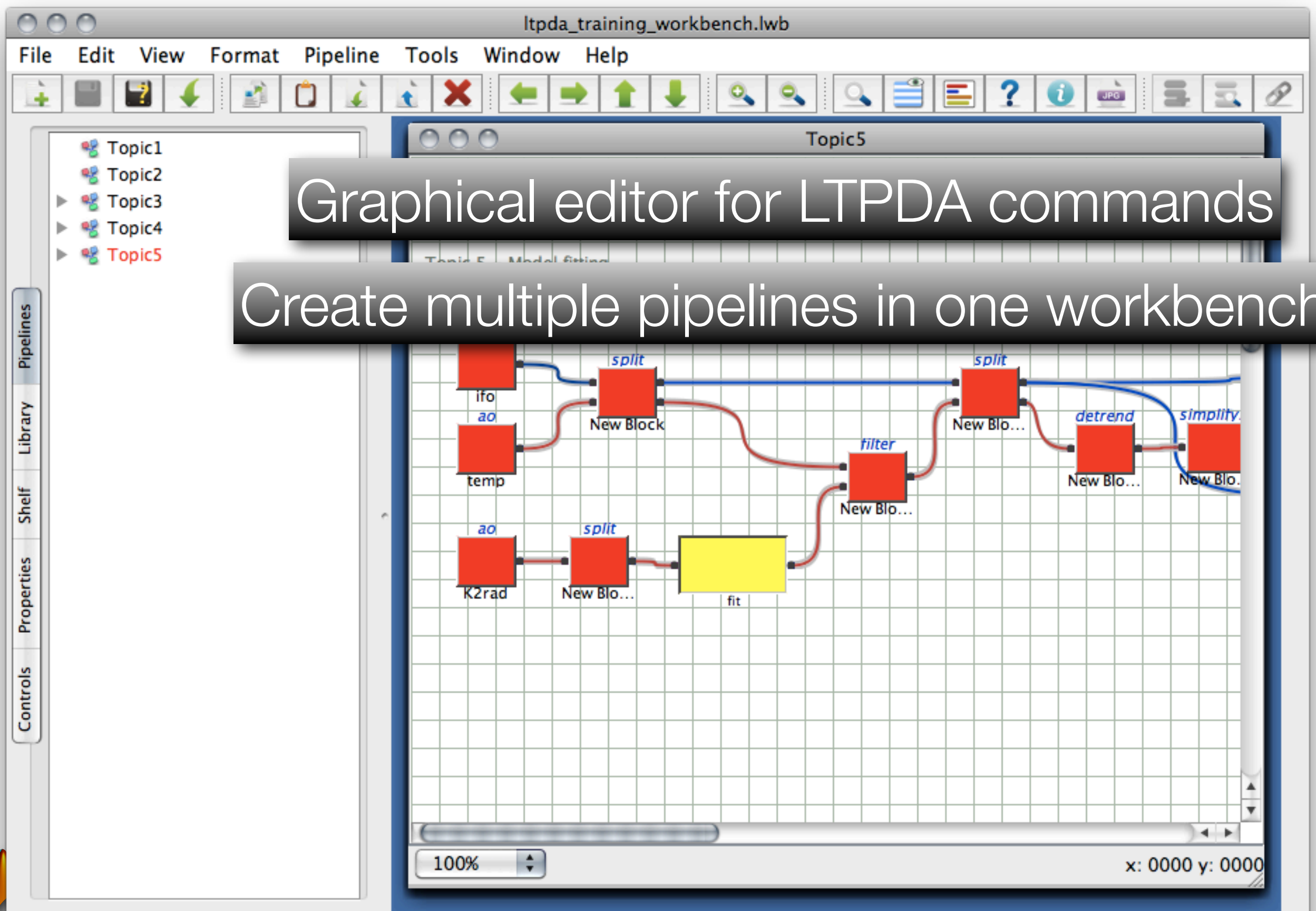
Introducing the workbench



Introducing the workbench



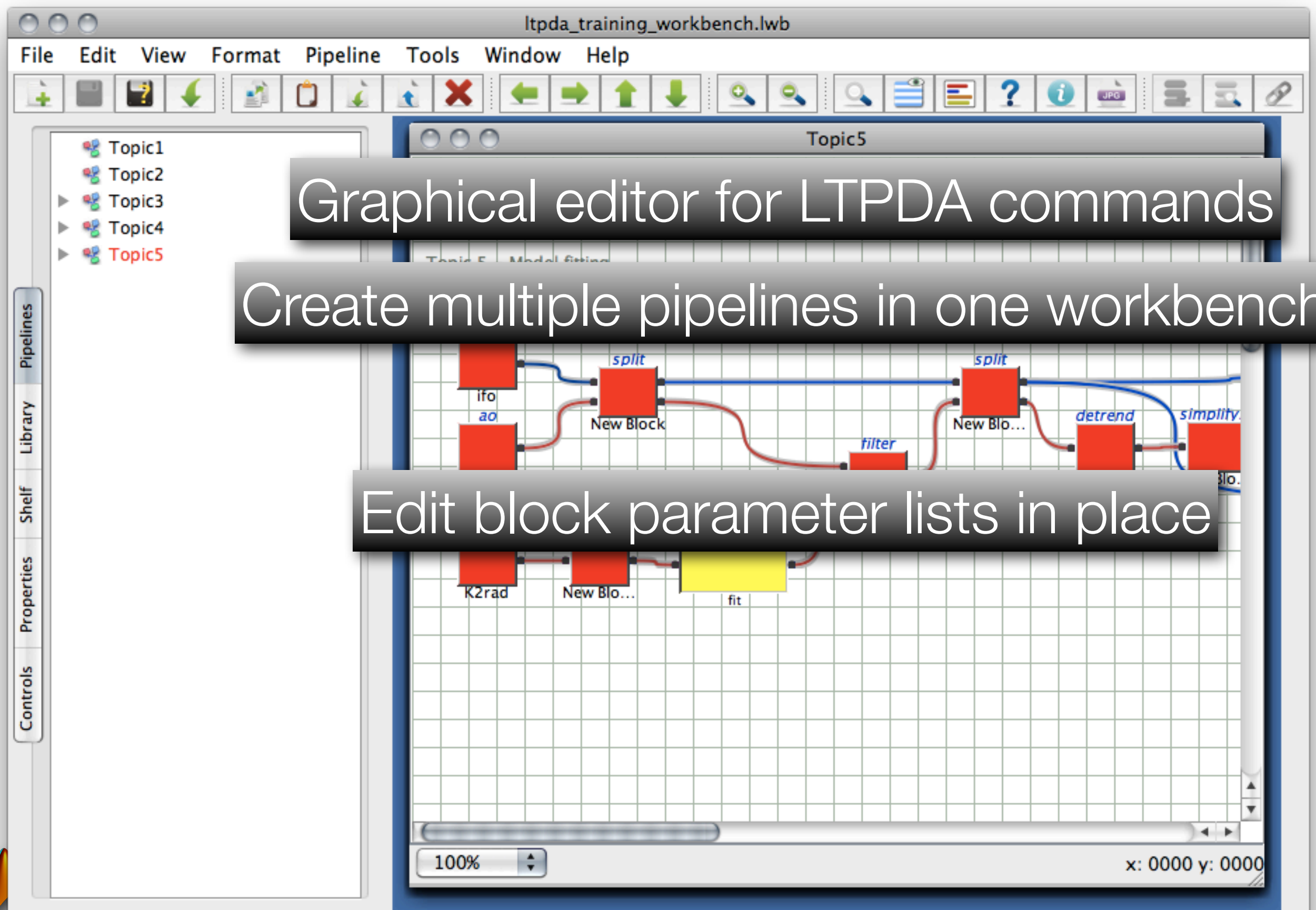
Introducing the workbench



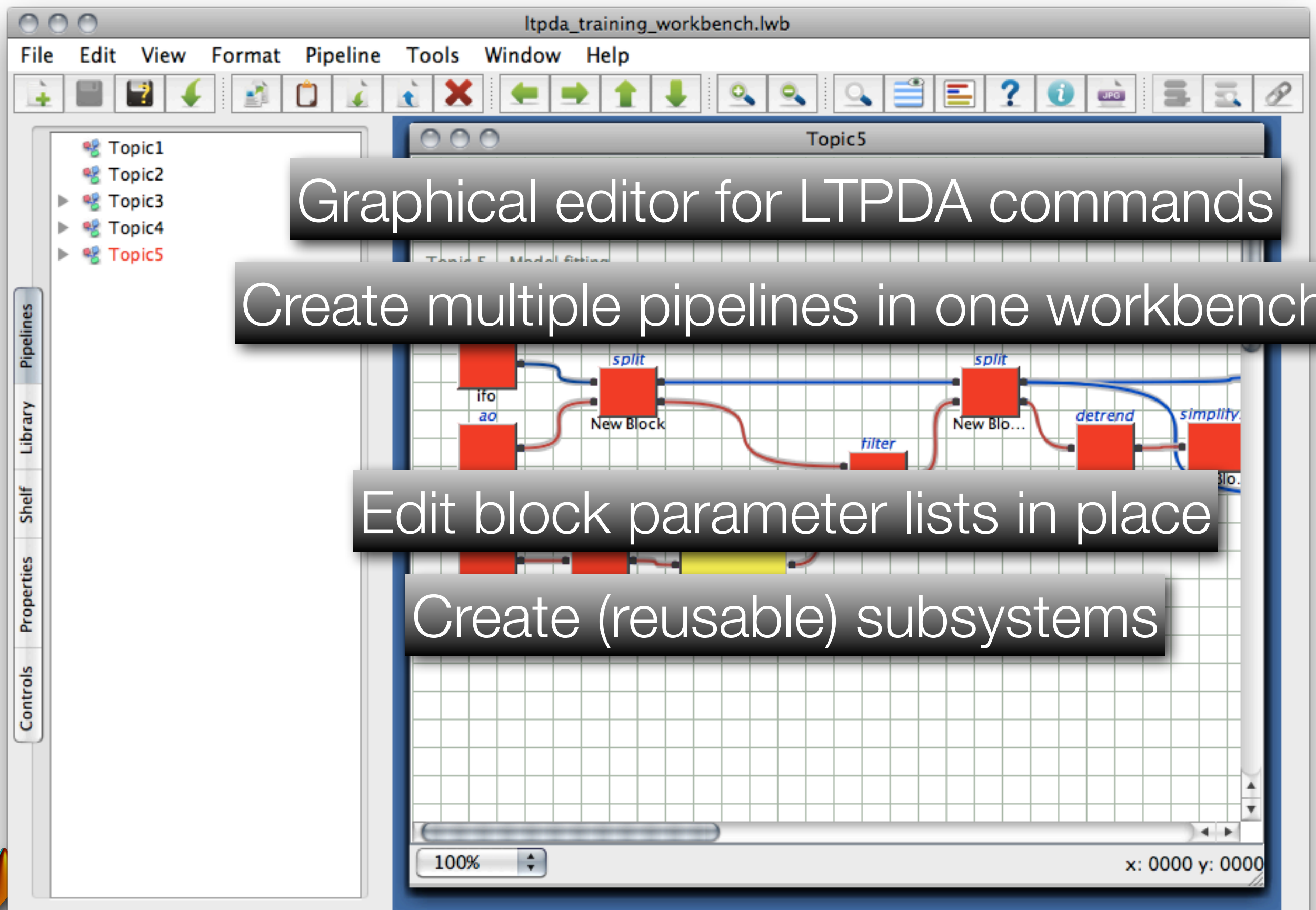
Graphical editor for LTPDA commands

Create multiple pipelines in one workbench

Introducing the workbench



Introducing the workbench



The screenshot shows a software window titled "ltpda_training_workbench.lwb". It has a menu bar with "File", "Edit", "View", "Format", "Pipeline", "Tools", "Window", and "Help". Below the menu is a toolbar with various icons for file operations, navigation, and editing. On the left, there is a sidebar with a "Pipelines" tab selected, showing a list of topics: "Topic1", "Topic2", "Topic3", "Topic4", and "Topic5" (highlighted in red). Below this are tabs for "Library", "Shelf", "Properties", and "Controls". The main workspace is a grid where a pipeline is being edited. It includes blocks labeled "ifo", "ao", "split", "New Block", "filter", "New Blo...", "detrend", and "simplify". Red and blue lines connect these blocks. At the bottom, there is a zoom slider set to "100%" and coordinates "x: 0000 y: 0000".

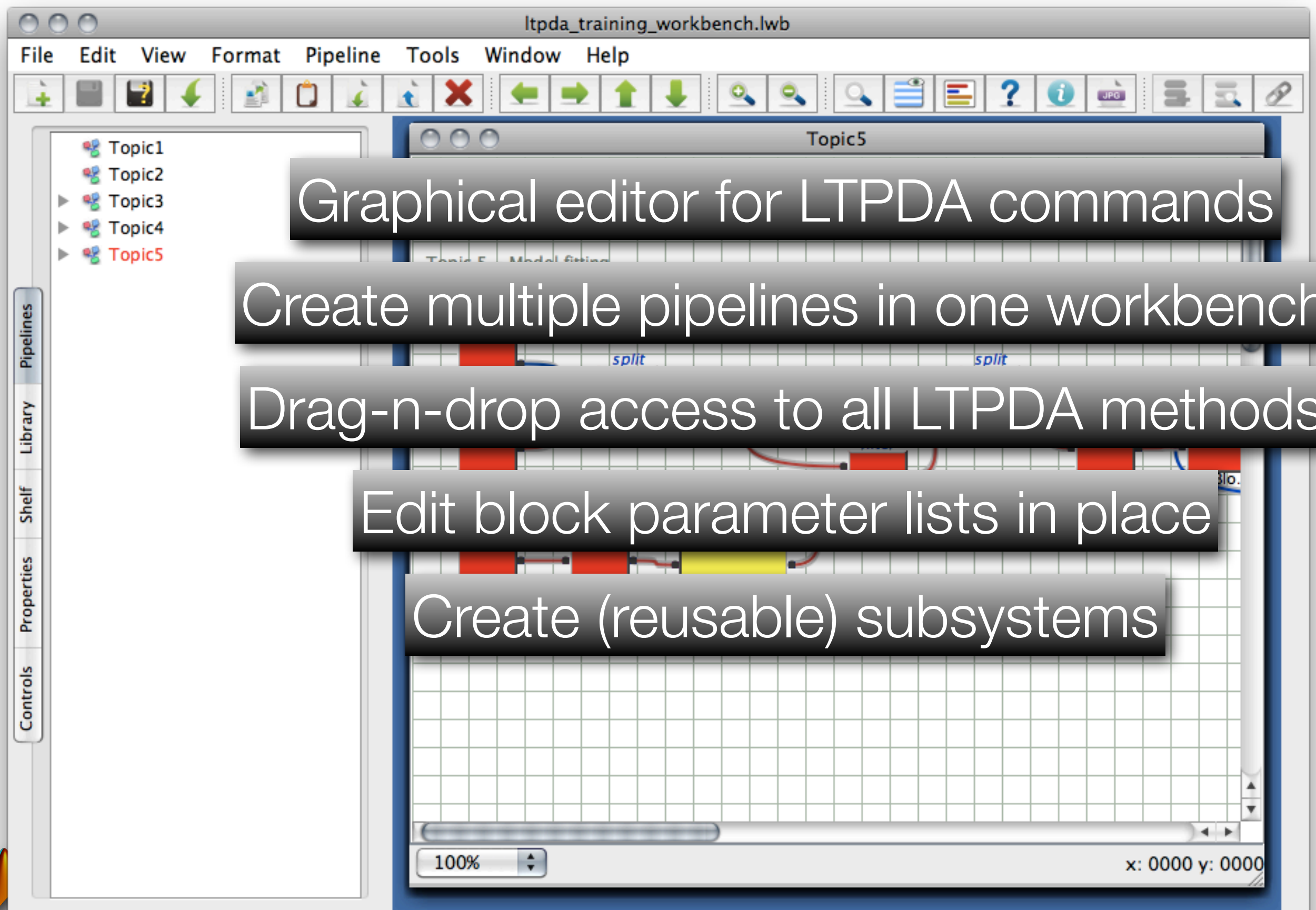
Graphical editor for LTPDA commands

Create multiple pipelines in one workbench

Edit block parameter lists in place

Create (reusable) subsystems

Introducing the workbench



Graphical editor for LTPDA commands

Create multiple pipelines in one workbench

Drag-n-drop access to all LTPDA methods

Edit block parameter lists in place



Create (reusable) subsystems

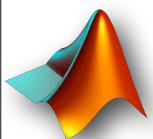
Parameter list editing

Current Parameters

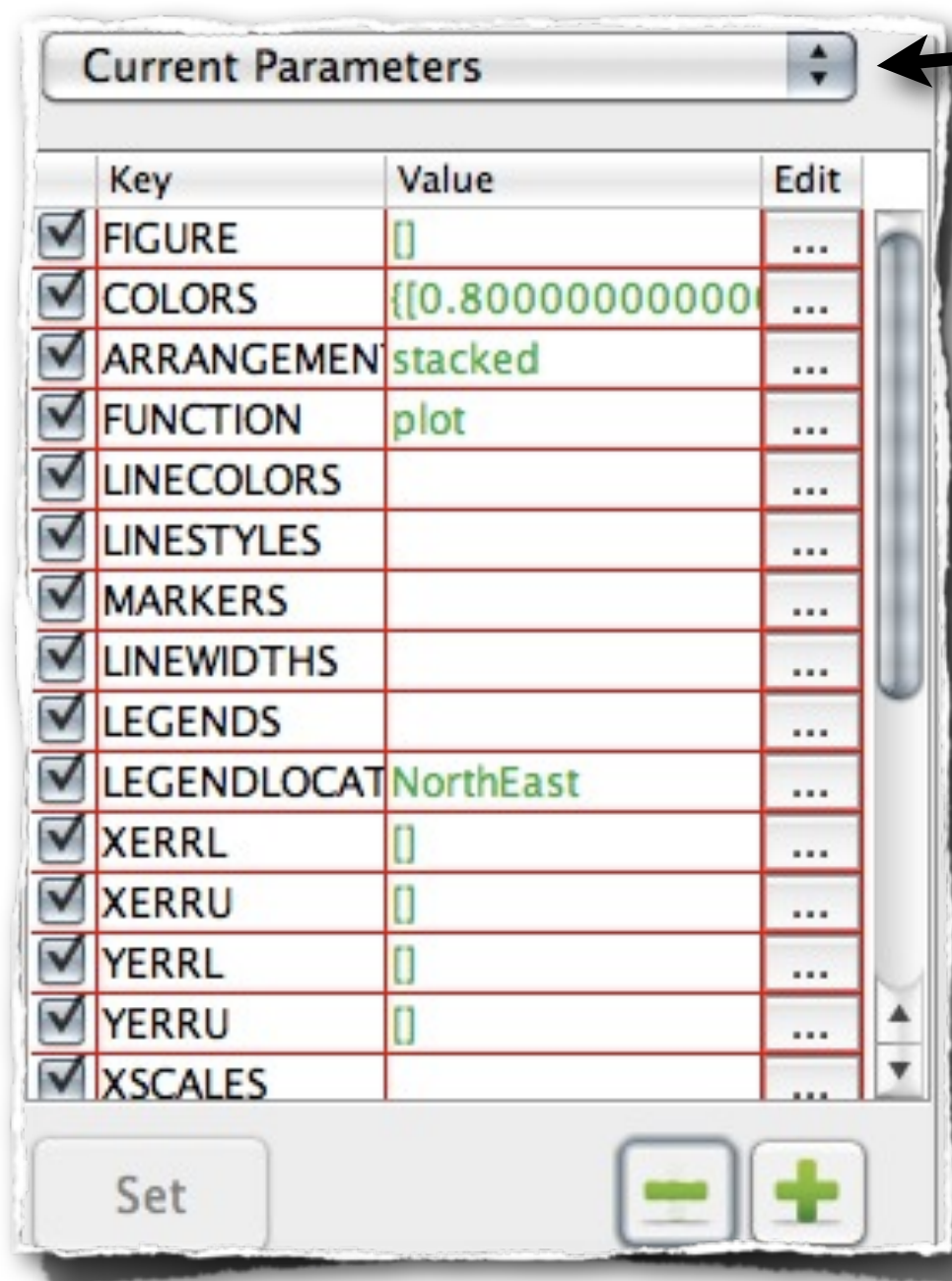
	Key	Value	Edit
<input checked="" type="checkbox"/>	FIGURE	<code>[]</code>	...
<input checked="" type="checkbox"/>	COLORS	<code>{{0.800000000000}}</code>	...
<input checked="" type="checkbox"/>	ARRANGEMENT	<code>stacked</code>	...
<input checked="" type="checkbox"/>	FUNCTION	<code>plot</code>	...
<input checked="" type="checkbox"/>	LINECOLORS		...
<input checked="" type="checkbox"/>	LINESTYLES		...
<input checked="" type="checkbox"/>	MARKERS		...
<input checked="" type="checkbox"/>	LINEWIDTHS		...
<input checked="" type="checkbox"/>	LEGENDS		...
<input checked="" type="checkbox"/>	LEGENDLOCAT	<code>NorthEast</code>	...
<input checked="" type="checkbox"/>	XERRL	<code>[]</code>	...
<input checked="" type="checkbox"/>	XERRU	<code>[]</code>	...
<input checked="" type="checkbox"/>	YERRL	<code>[]</code>	...
<input checked="" type="checkbox"/>	YERRU	<code>[]</code>	...
<input checked="" type="checkbox"/>	XSCALES		...

Set

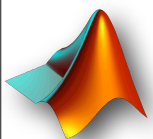





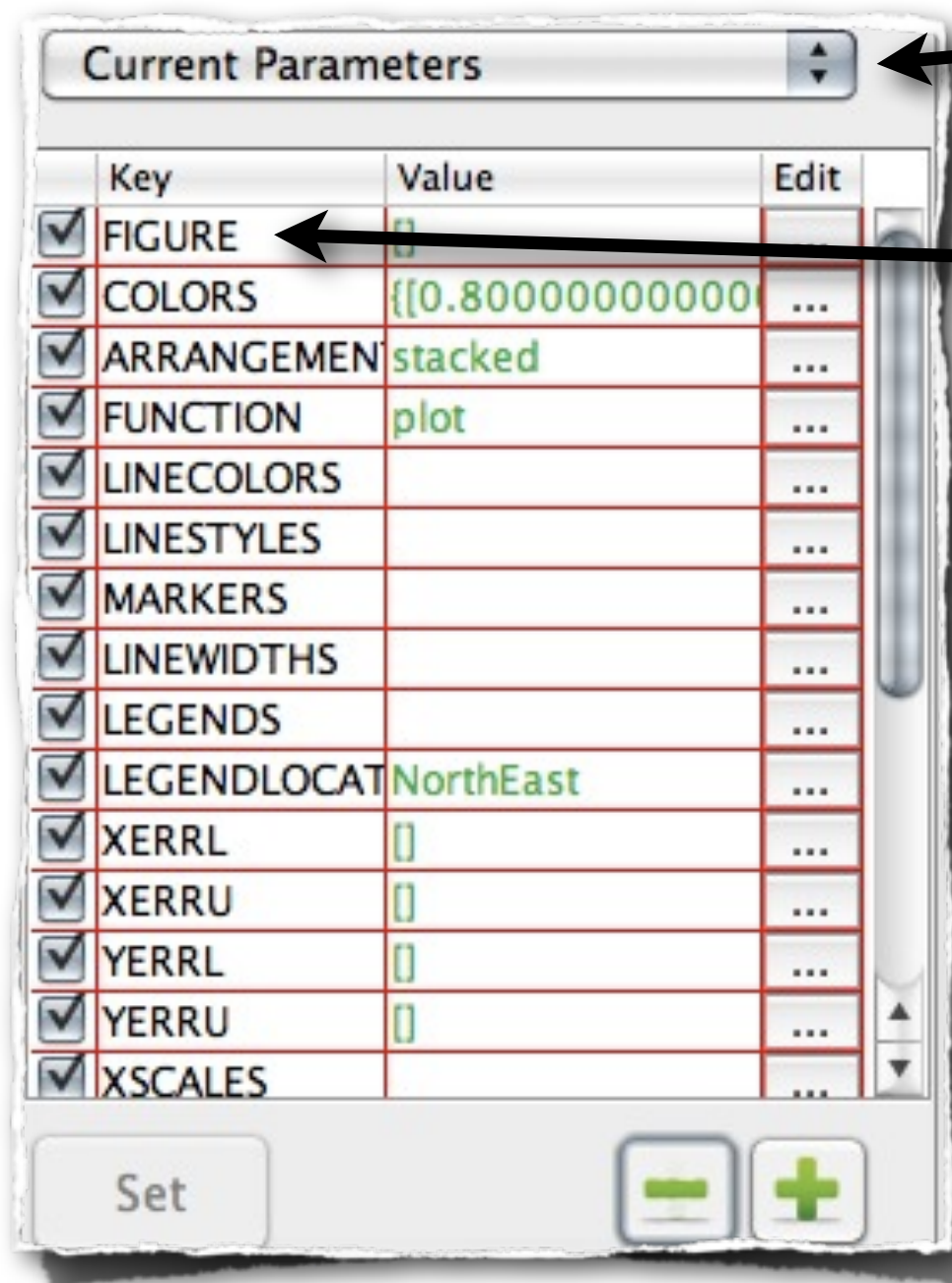
Parameter list editing



Different parameter sets

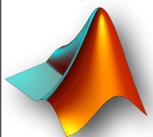


Parameter list editing

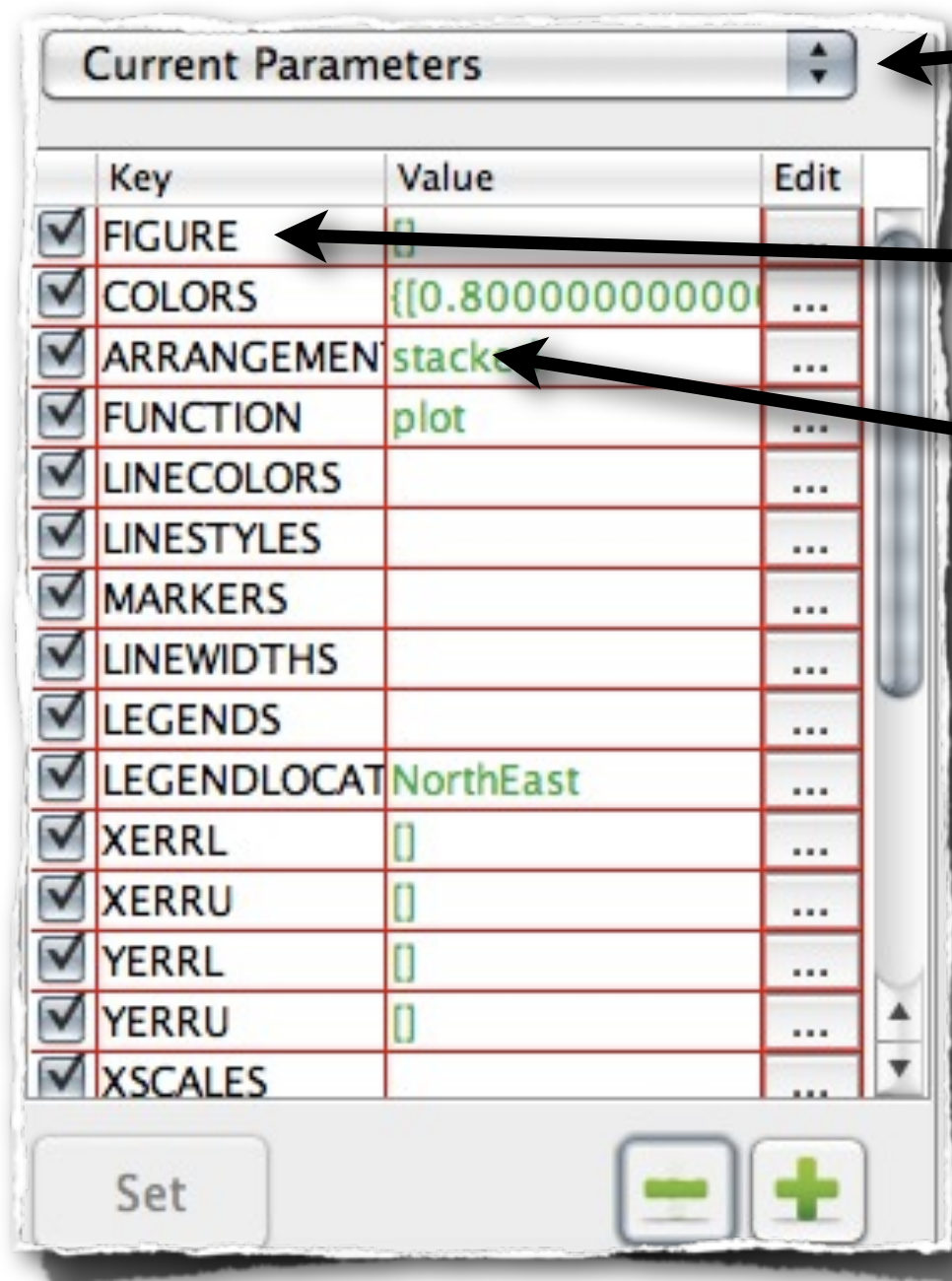


Different parameter sets

Parameter 'key'



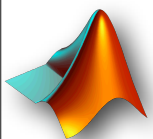
Parameter list editing



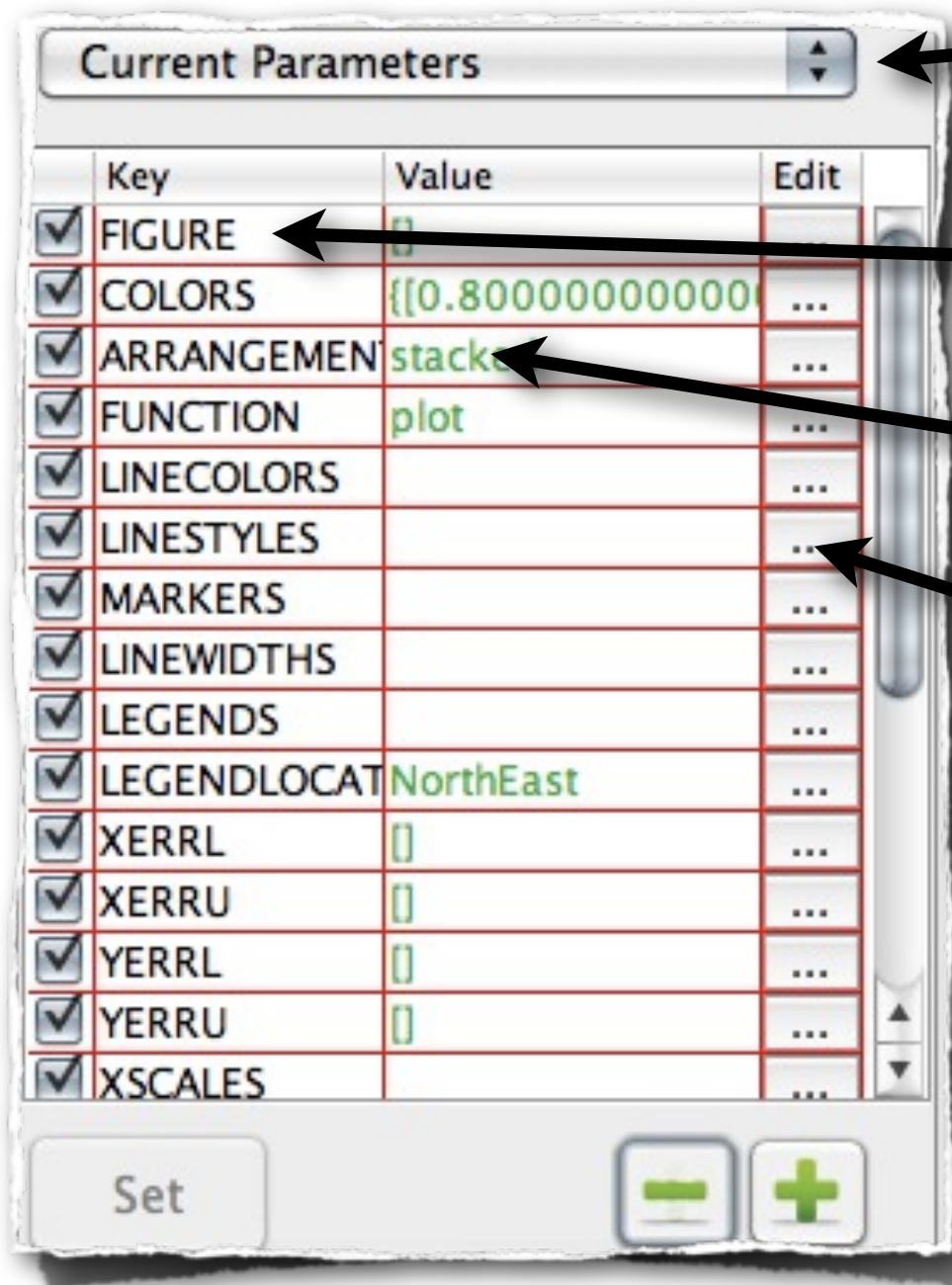
Different parameter sets

Parameter 'key'

Parameter 'value'



Parameter list editing

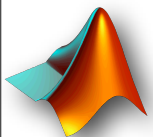


Different parameter sets

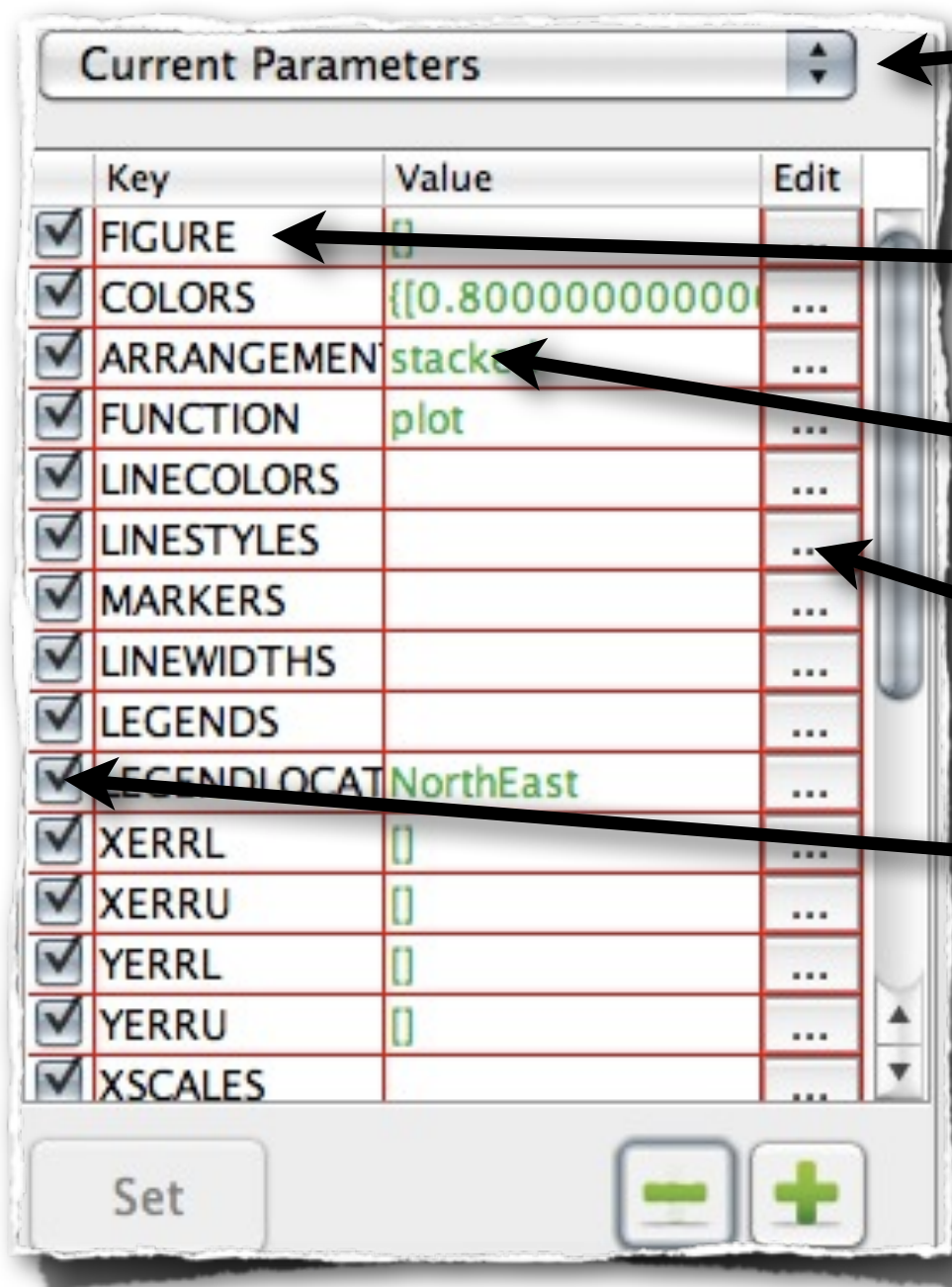
Parameter 'key'

Parameter 'value'

Open 'special' editor



Parameter list editing



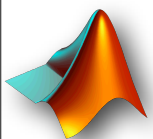
Different parameter sets

Parameter 'key'

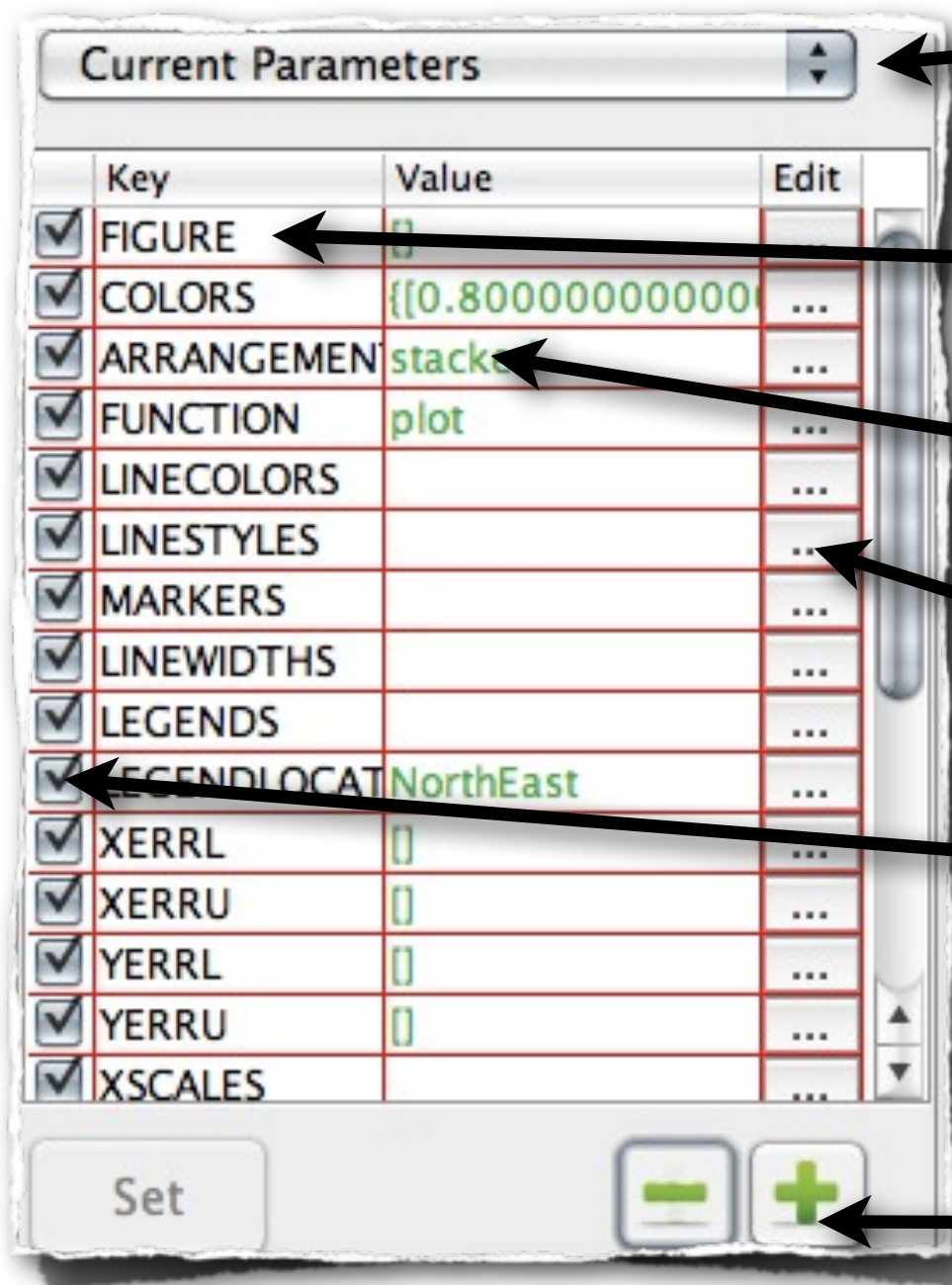
Parameter 'value'

Open 'special' editor

Activate/de-activate a parameter



Parameter list editing



Different parameter sets

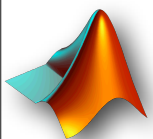
Parameter 'key'

Parameter 'value'

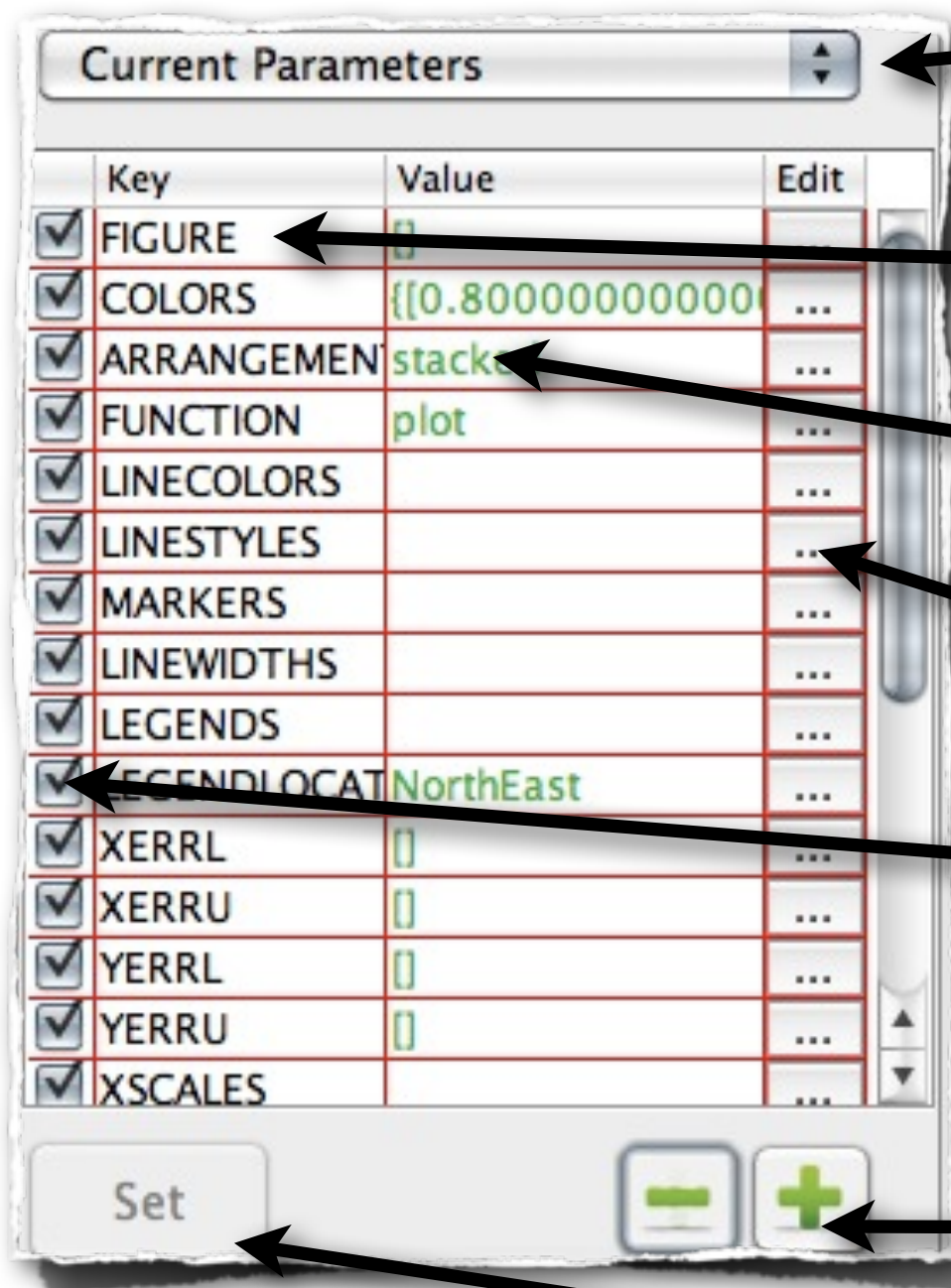
Open 'special' editor

Activate/de-activate a parameter

Add/remove a parameter



Parameter list editing



Different parameter sets

Parameter 'key'

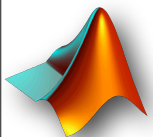
Parameter 'value'

Open 'special' editor

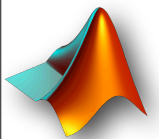
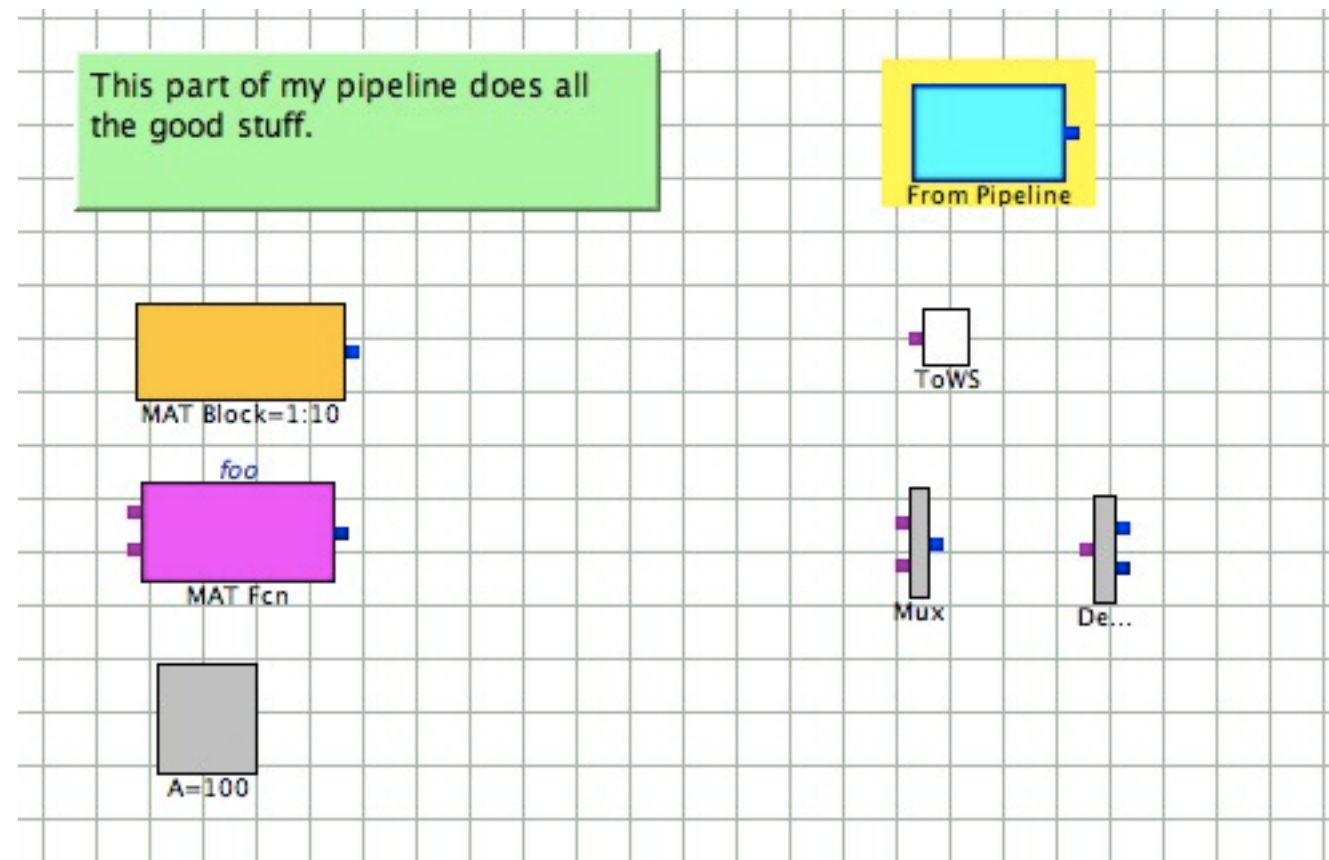
Activate/de-activate a parameter

Add/remove a parameter

'Set' a parameter set



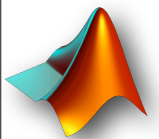
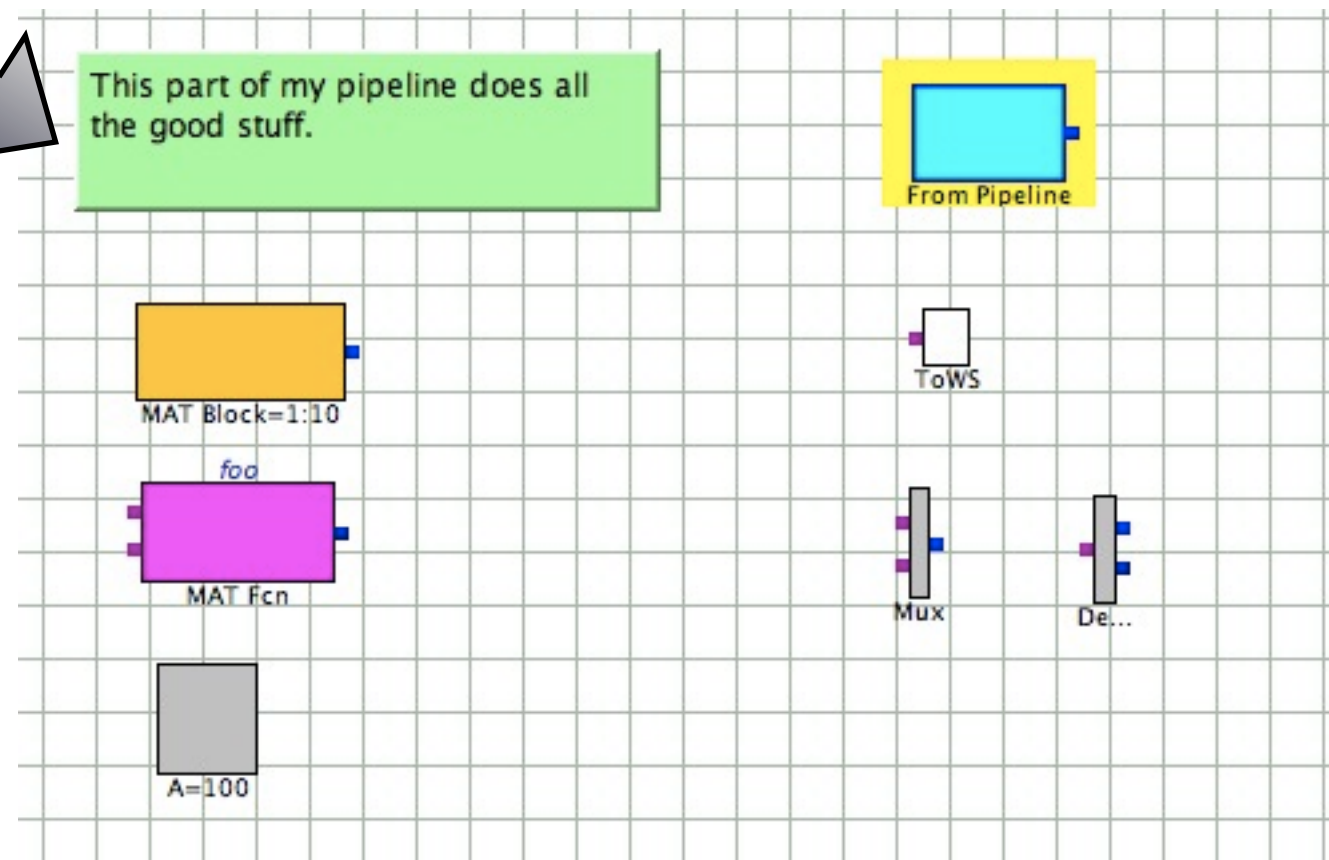
Special blocks



Special blocks



Annotation

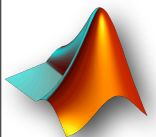
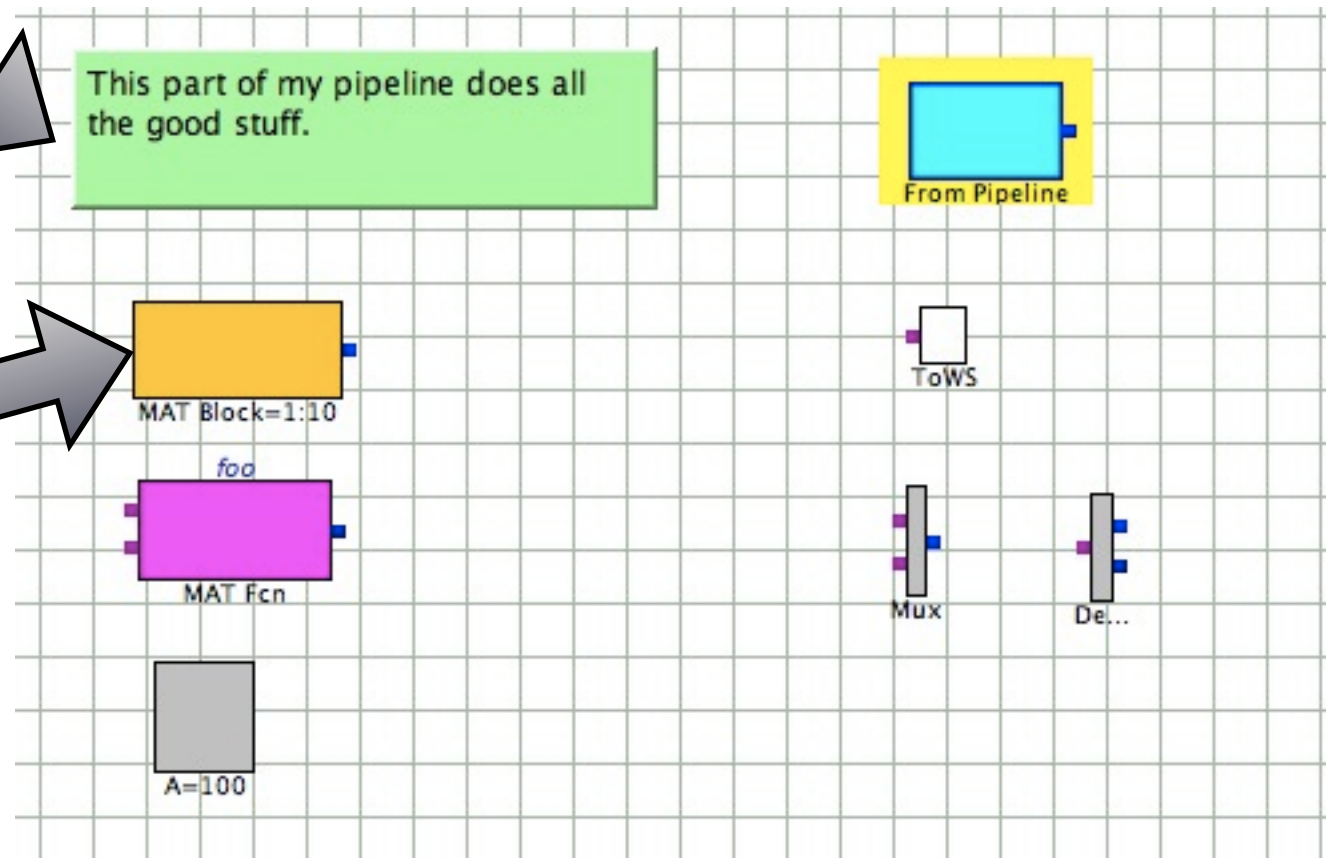


Special blocks

Annotation

This part of my pipeline does all the good stuff.

MATLAB Expressions



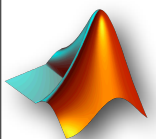
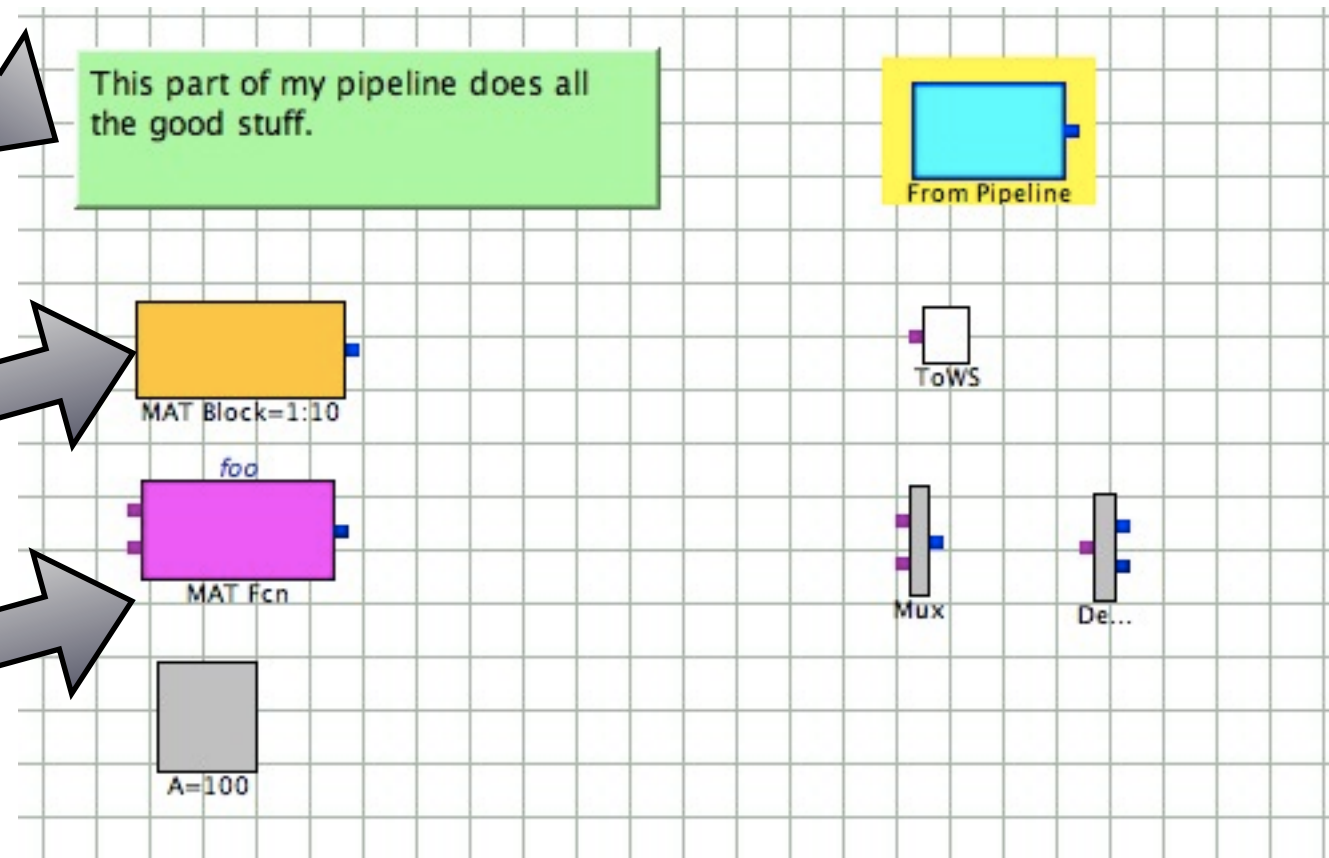
Special blocks

Annotation

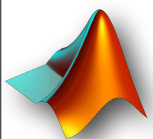
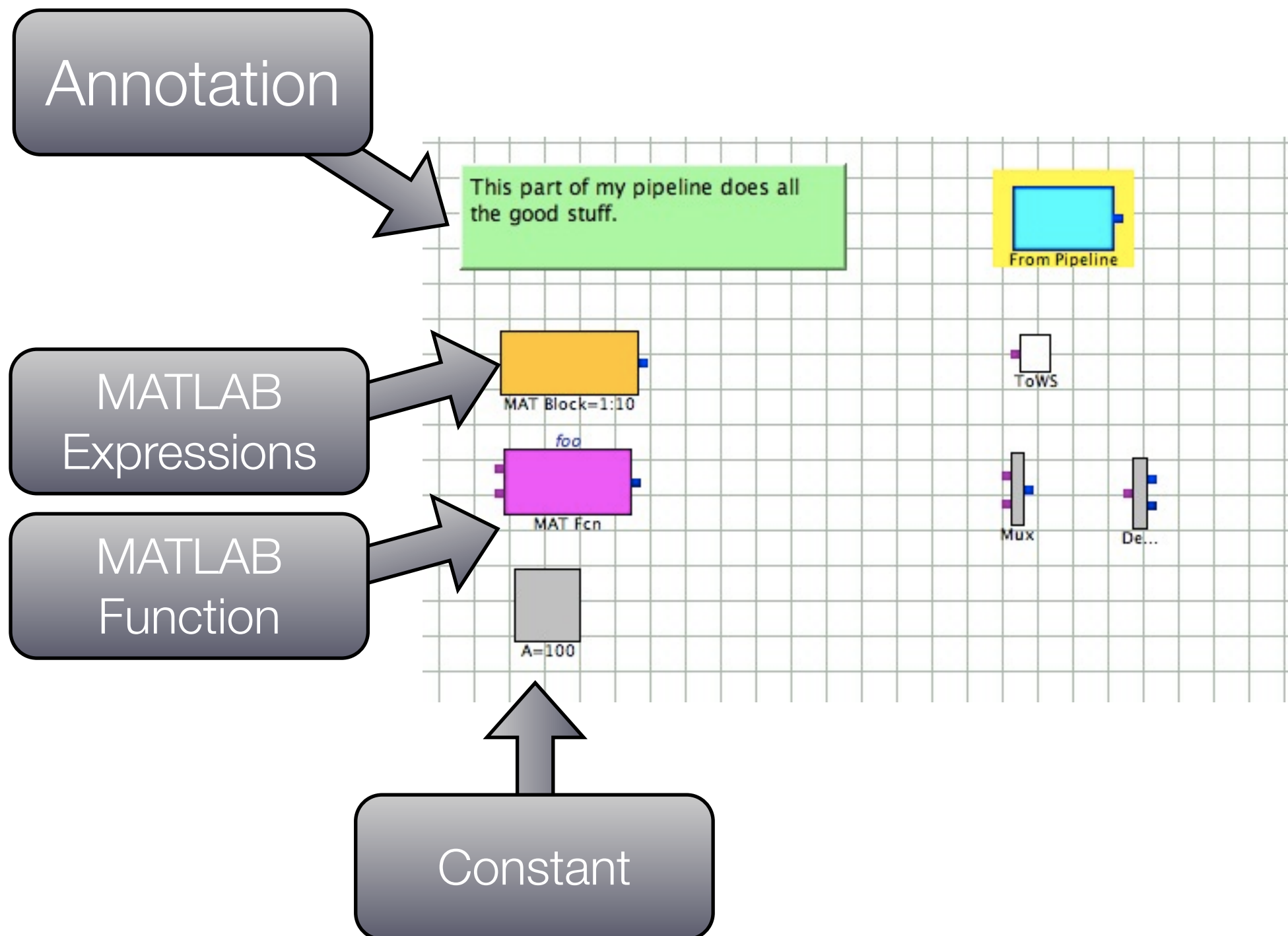
This part of my pipeline does all the good stuff.

MATLAB Expressions

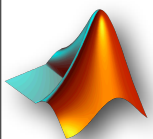
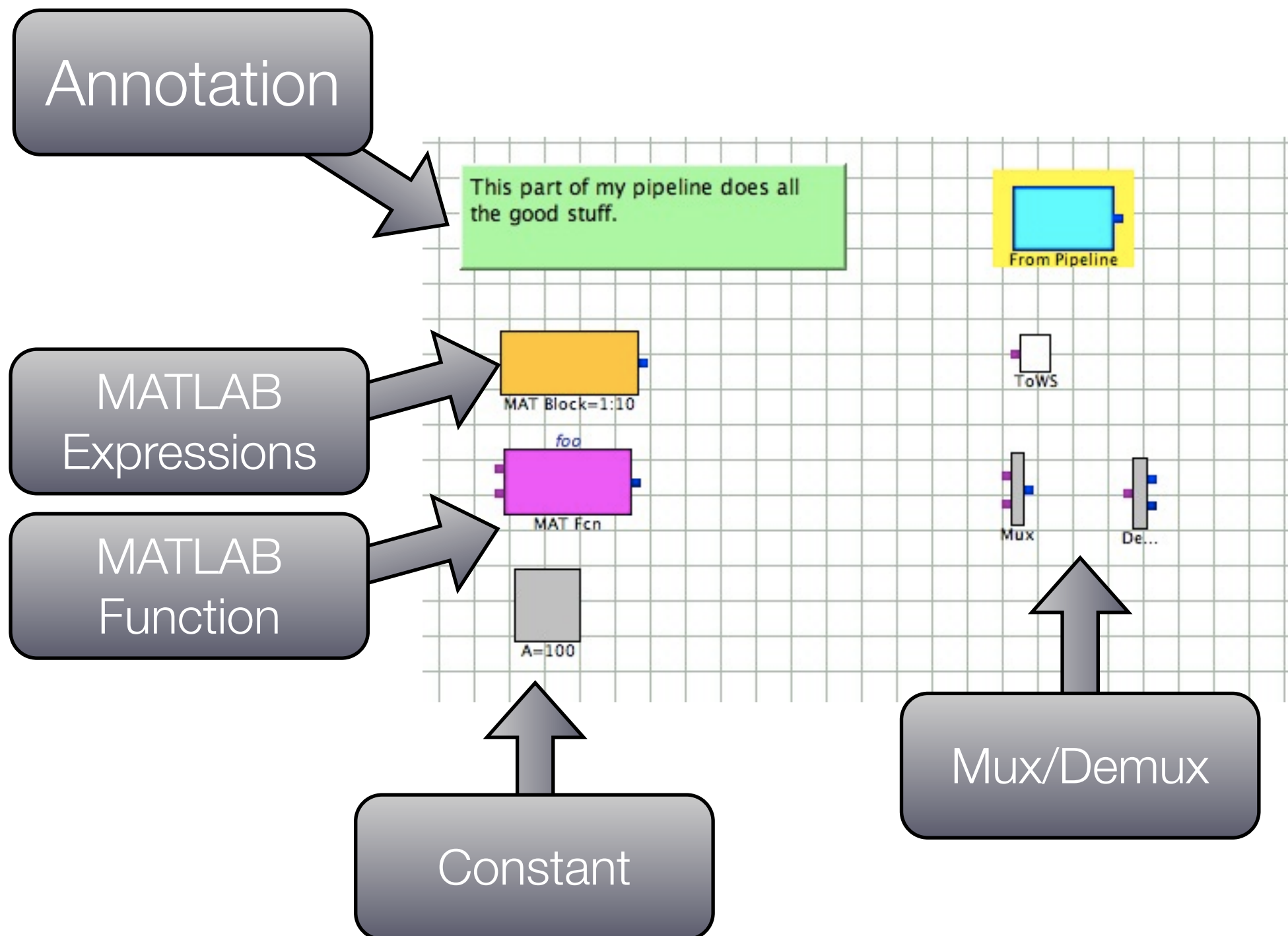
MATLAB Function



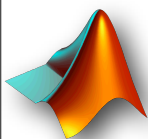
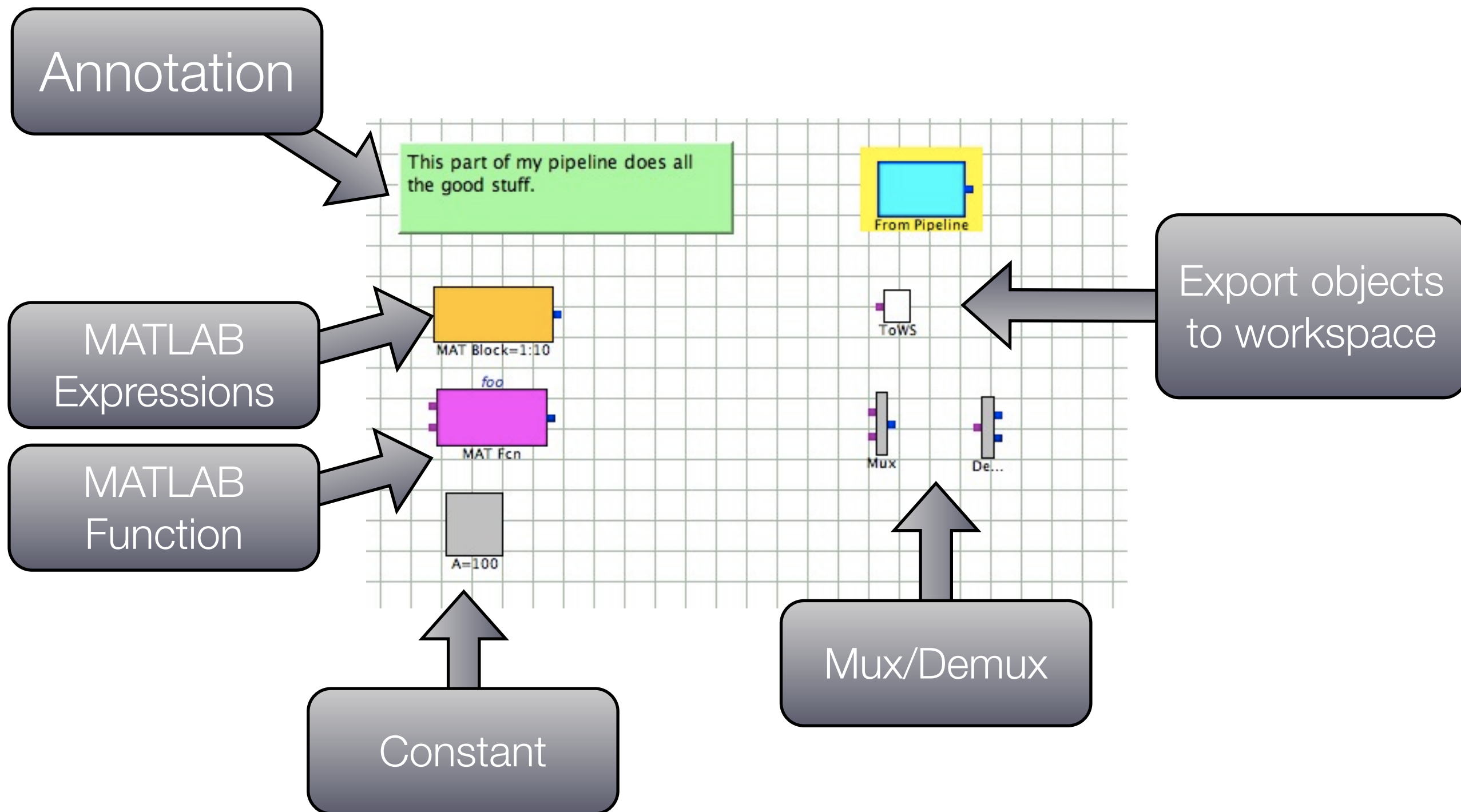
Special blocks



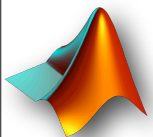
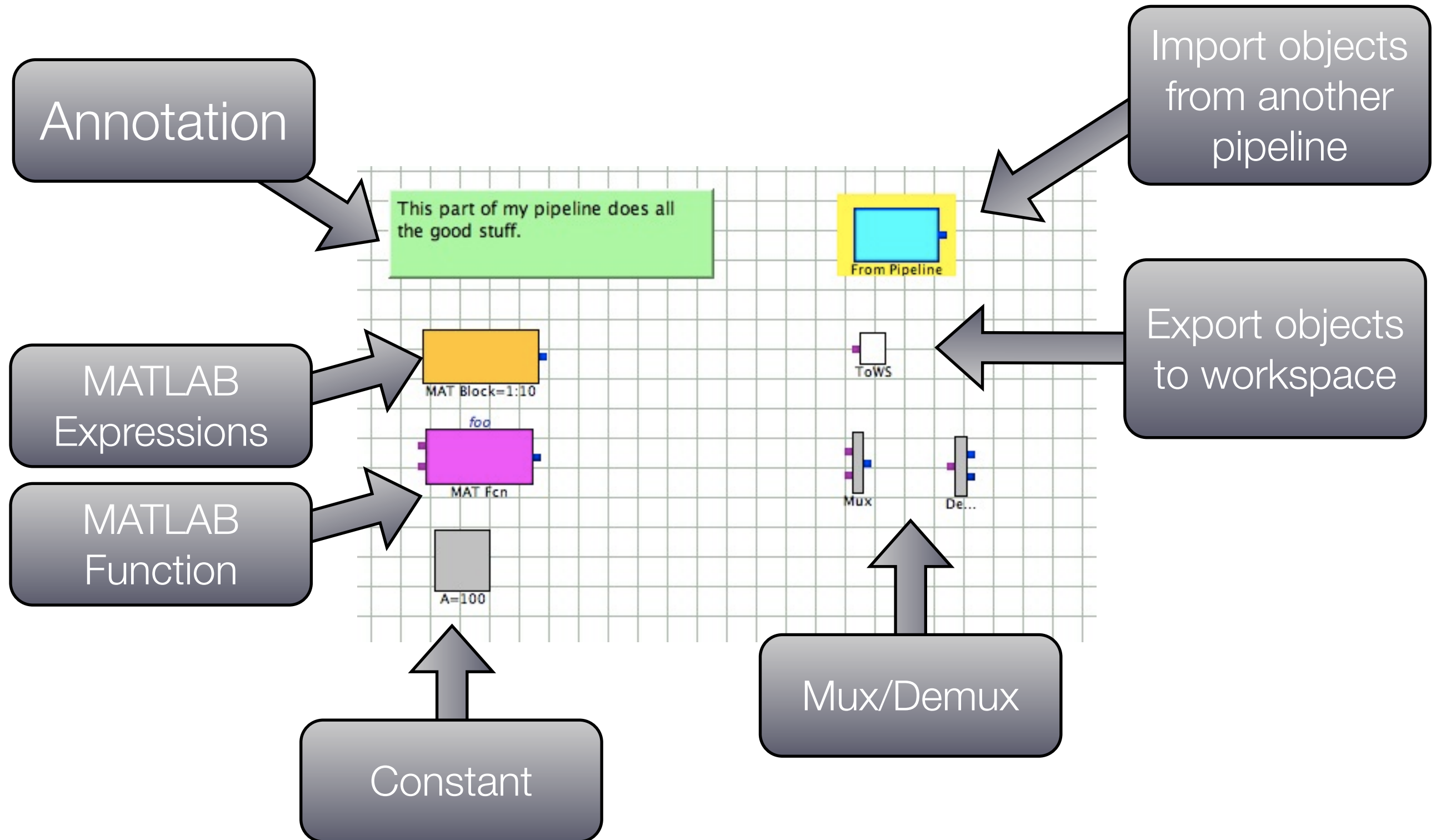
Special blocks



Special blocks



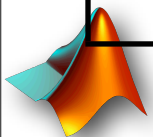
Special blocks



Shortcut keys



<code>ctrl-s</code>	Save workbench
<code>ctrl-n</code>	New pipeline in current workbench
<code>ctrl-c, ctrl-v, ctrl-d</code>	copy, paste, duplicate
<code>ctrl-t</code>	comment-out block(s)
<code>ctrl-b</code>	quick-block dialog
<code>ctrl-i</code>	edit canvas info
<code>ctrl-f</code>	find blocks in workbench
<code>shift-F1</code>	help on selected block

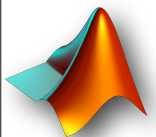


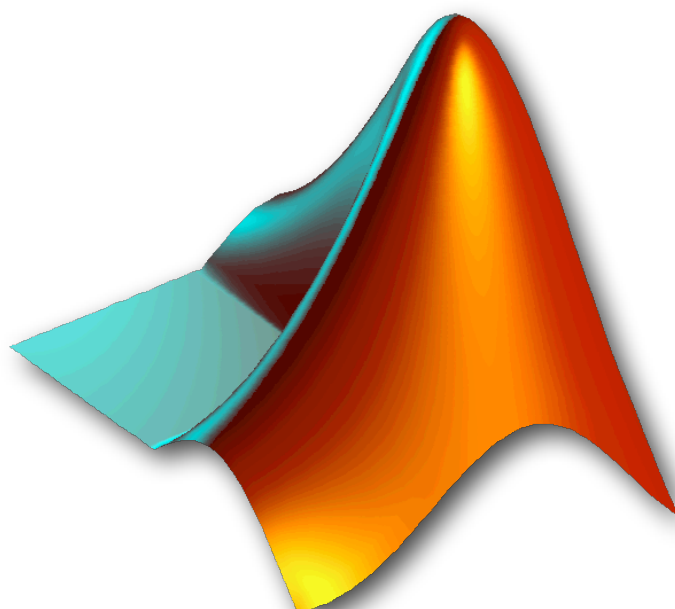
Workbench Fix



- copy reset.m to

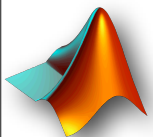
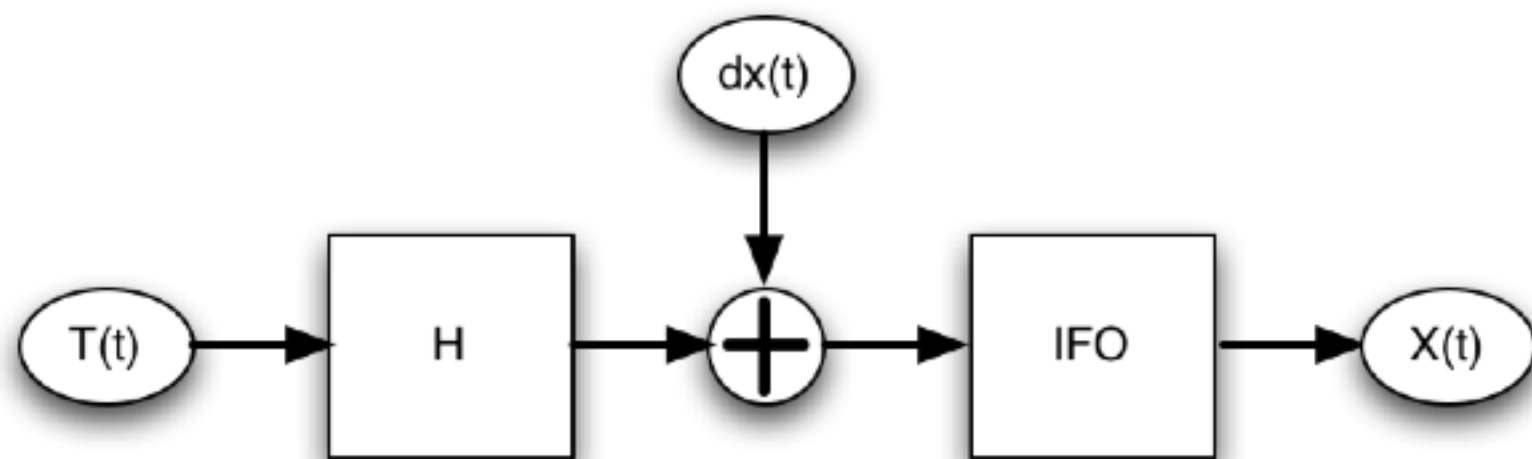
<somewhere>/ltpda_toolbox_2.3/ltpda/classes/
@LTPDAworkbench/reset.m

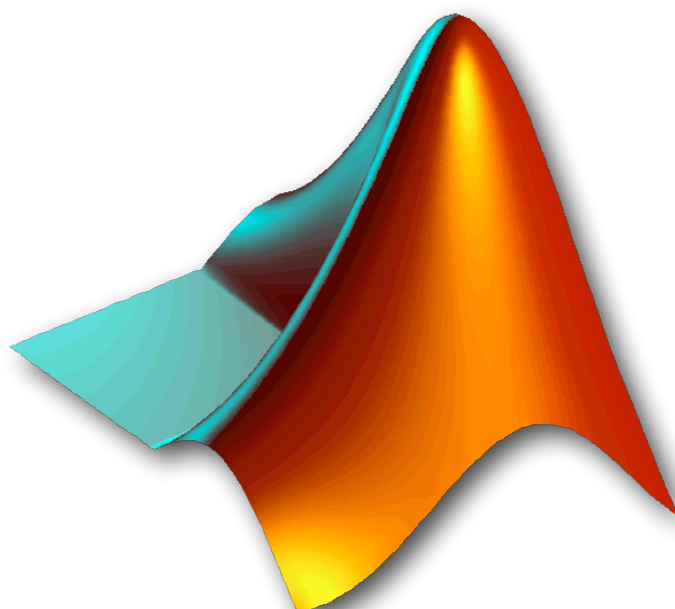




Interferometer-Temperature example

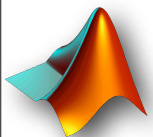
- We have a data analysis exercise which will develop fully over the course of the training session
 - This is the first part: reading and preparing the data
- Work through help section
 - Topic 1
 - IFO/Temperature Example - Introduction





Topic 2: Pre-processing data

- Why?
 - data preparation for further analysis
- LTPDA contains a bunch of functions for
 - resampling data
 - interpolation of data
 - de-trending data
 - noise whitening
 - data selection



Look in the help!

A screenshot of the LTPDA Toolbox help documentation. The left pane shows a tree view with "Signal Pre-processing in LTPDA" selected. The right pane shows the "Signal Pre-processing in LTPDA" page, which includes a description of the pre-processing functions and a list of links to specific tools. The bottom of the right pane shows a navigation bar with "Converting models to digital filters" and "Downsampling data".

Search

Contents Search Results

LTPDA Toolbox

- ▶ Getting Started with the LTPDA Toolbox
- ▶ Examples
- ▶ Introducing LTPDA Objects
- ▶ Parameter Lists
- ▶ Simulation/modelling
- ▼ **Signal Pre-processing in LTPDA**
 - Downsampling data
 - Upsampling data
 - Resampling data
 - Interpolating data
 - Spikes reduction in data
 - Data gap filling
 - Noise whitening
- ▶ Signal Processing in LTPDA
- ▶ Graphical User Interfaces in LTPDA
- ▶ Working with an LTPDA Repository
- ▶ Class descriptions
- ▶ Functions - By Category
- ▼ LTPDA Training Session 1

LTPDA Toolbox [contents](#)

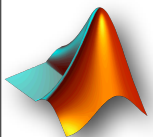
Signal Pre-processing in LTPDA

Signal pre-processing in LTPDA consists on a set of functions intended to pre-process data prior to further analysis. Pre-processing tools are focused on data sampling rates manipulation, data interpolation, spike cleaning and gap filling functions.

The following pages describe the different pre-processing tools available in the LTPDA toolbox:

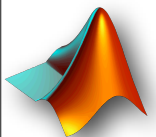
- [Downsampling data](#)
- [Upsampling data](#)
- [Resampling data](#)
- [Interpolating data](#)
- [Spikes reduction in data](#)
- [Data gap filling](#)
- [Noise Whitening](#)

◀ Converting models to digital filters Downsampling data ▶



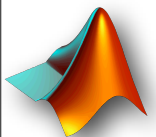
Resampling

- Integer factor
 - Down-sample (ao/downsample)
 - for example, to reduce data load
 - Up-sample (ao/upsample)
 - for example, to match sample rates, do 'better' filtering
- Re-sample (ao/resample)
 - $fs_{out} = P/Q * fs_{in}$ (P and Q are integers)

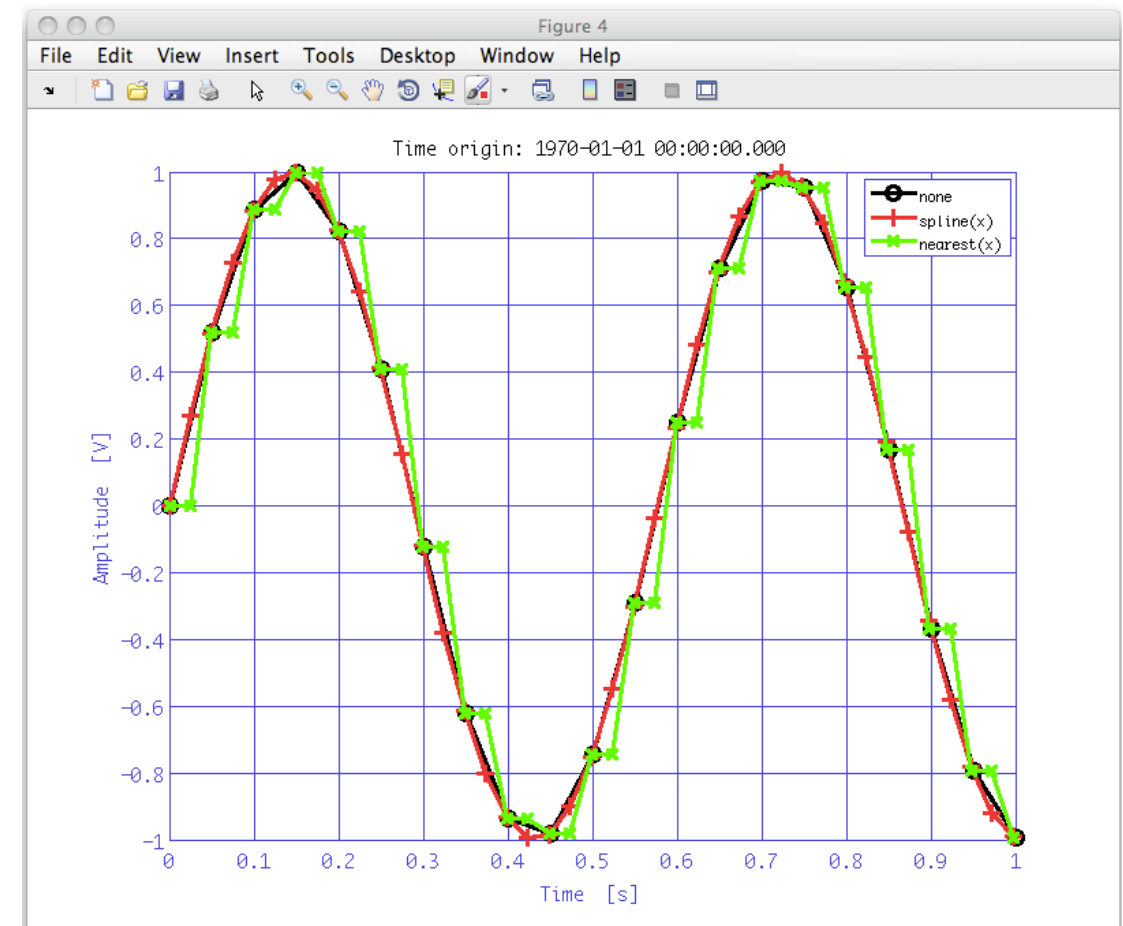


Topic 2 - Exercises 1,2,3

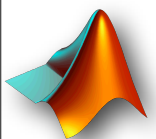
- Open MATLAB documentation
 - In the MATLAB terminal
 - `>> doc`
 - “Help -> Product Help>”
- work through
 - LTPDA Toolbox - LTPDA Training Session 1 Topic 2
 - Downsampling
 - Upsampling
 - Resampling



Interpolation



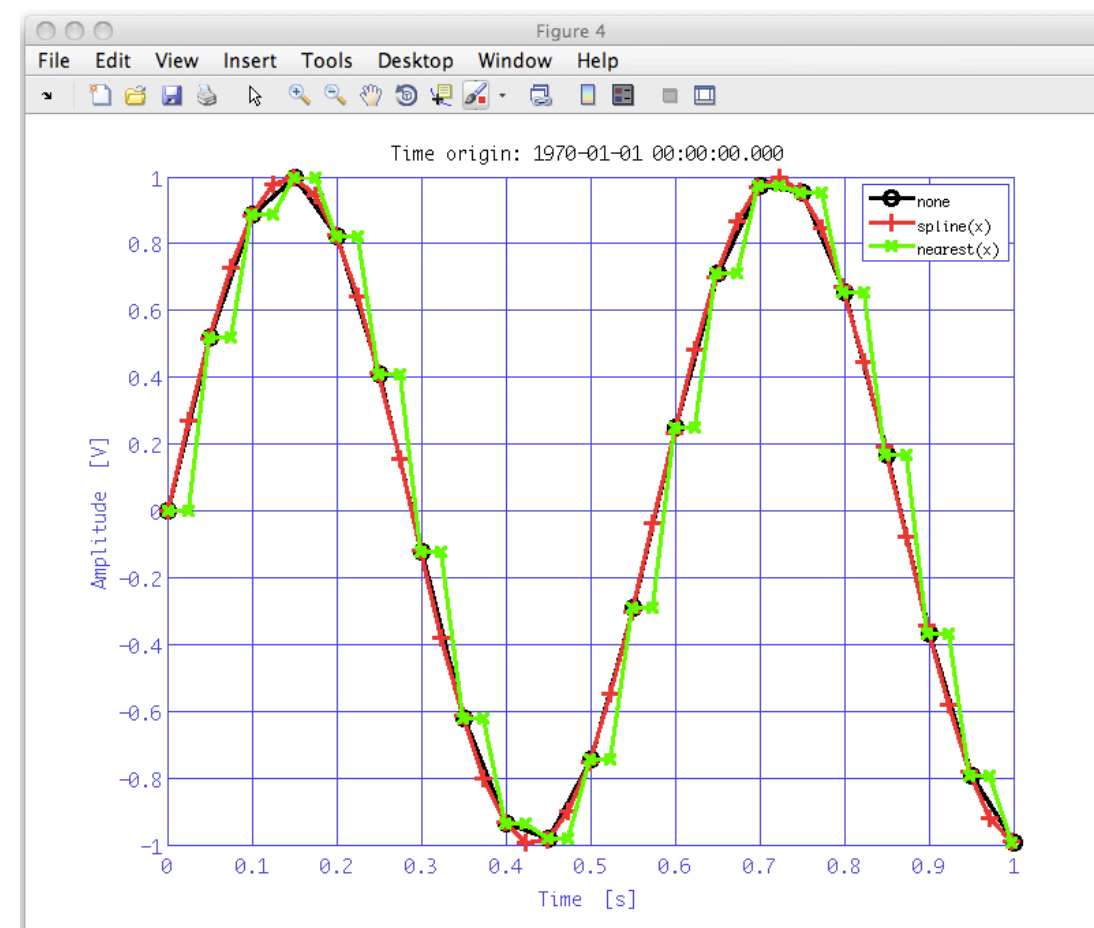
work through:
Topic 2- Interpolation



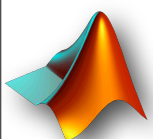
Interpolation



'vertices'	new time grid
interpolation methods	
'linear'	linear interpolation
'spline'	spline interpolation
'cubic'	cubic interpolation
'nearest'	nearest neighbour



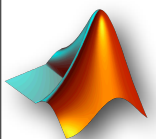
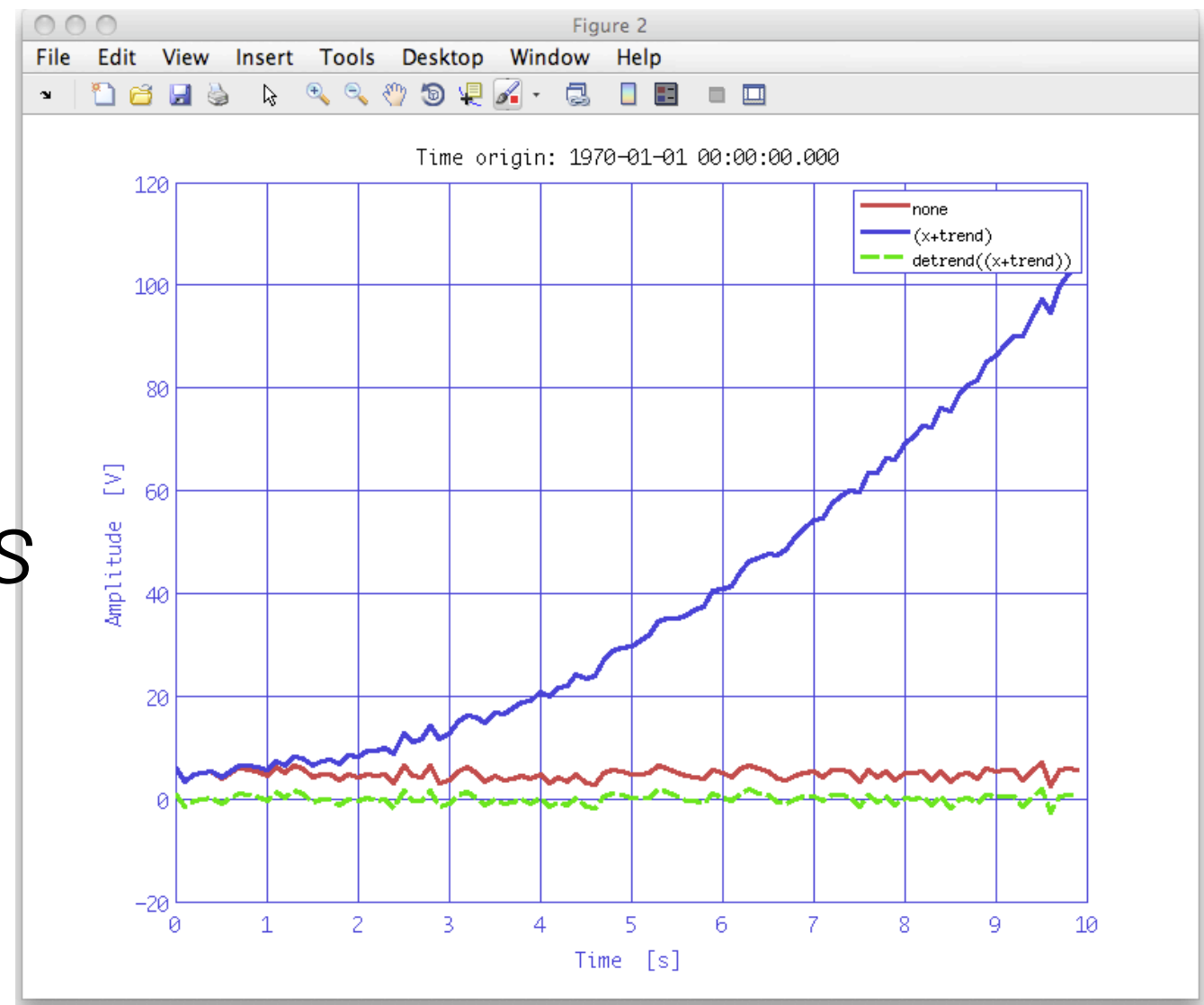
work through:
Topic 2- Interpolation



Detrending data

- Remove trends by
 - subtracting polynomial fit from data
- ao/detrend calls MATLABs polyfit

work through:
Topic 2- Removing trends

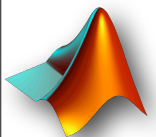


Whitening



- The LTDA Toolbox offers various ways to whiten your data
 - with a known filter
 - build filter and apply it to your data
 - with a known model of spectral content
 - use `whiten1D`
 - for single, uncorrelated data streams
 - `whiten2D`
 - for a pair of correlated data streams
 - without model (Exercise)
 - let `whiten1D` fit a model to the spectrum of your data

work through
Topic 2 whitening

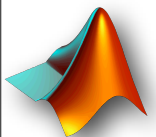


Select & find/ split & join

- Chose the samples you want to analyse
 - find/select data samples by its properties
 - sample numbers - 'select'
 - query for x and y values - 'find'
 - split data by
 - intervals, times, frequencies, samples
- Group of functions helps you to
 - for find and select exactly the data you want split your data into pieces and eventually
 - join them back together

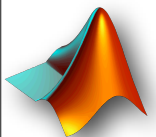
work through

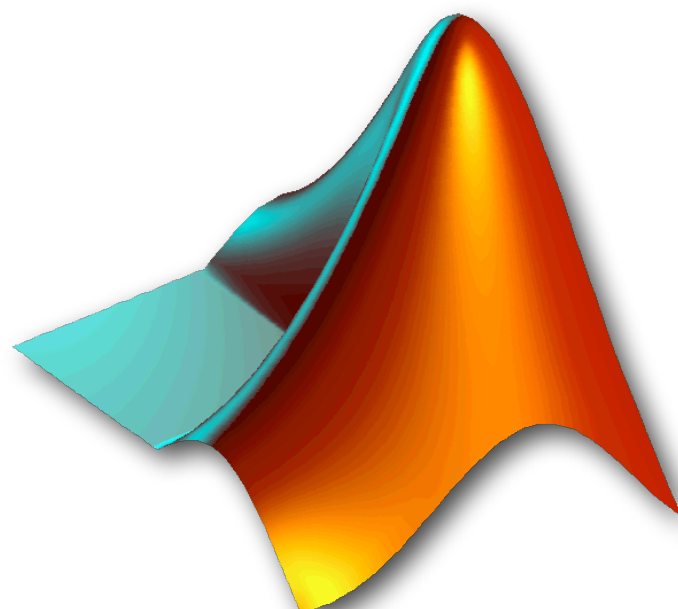
Topic 2 - select and find, split and join



Pre-processing the IFO/Temp data

- There is a useful function which ‘cleans up’ the data
 - `consolidate`
- `consolidate` fixes our two data streams such that
 - they start at the same time
 - they have the same sampling rate
 - they are evenly sampled on the same grid
- Work through
 - LTPDA Toolbox - LTPDA Training Session 1, Topic 2
 - IFO/Temp example





Power Spectral Density Estimation (1)

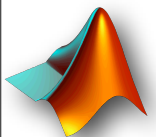


Definition:

$$P_{xx}(f) = \frac{1}{f_s} \sum_{m=-\infty}^{+\infty} R_{xx}(m) \exp(-2\pi i \cdot f \cdot m / f_s)$$

Estimates the one-sided PSD:

$$\tilde{P}_{xx}(f) = \begin{cases} 0, & -\frac{f_s}{2} \leq f \leq 0 \\ 2P_{xx}(f), & 0 \leq f \leq \frac{f_s}{2} \end{cases}$$



Power Spectral Density Estimation (2)



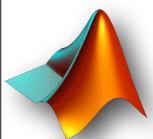
- The PSD at each frequency is estimated via the Welch method:
 - Given a discretized signal $x[n]$ of length N , Data are divided into segments of length L and multiplied by a window
 - this also reduces the edge-effects (simulating a periodic sequence)
 - The PSD at each frequency f is estimated as:

$$\hat{P}_{xx}(f) = \frac{|X_L|^2}{f_s \cdot L \cdot U}$$

- where

$$x_L[n] = x[n]w[n]$$

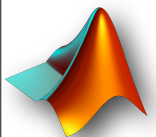
$$x_L(f) = \sum_{n=0}^{L-1} x_L[n] \exp\left(-2\pi i \cdot \frac{f \cdot n}{f_s}\right) \quad U = \frac{1}{L} \sum_{n=0}^{L-1} |w(n)|^2$$



Power Spectral Density Estimation (3)



- Methods:
 - ao/psd: linear frequency scale
 - ao/lpsd: log-frequency scale
 - specwin: implements spectral windows

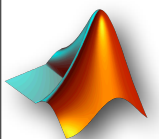
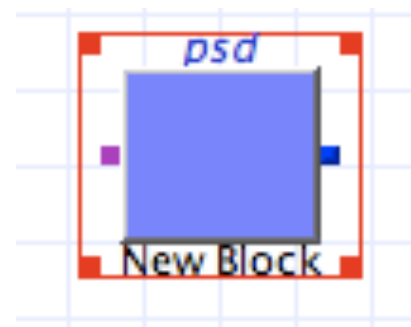


Power Spectral Density Estimation (4)



- a = [ao with time-series data]
- $S = a.\text{psd}(\text{plist}(\text{'win'}, \text{win}, \dots \text{'nfft'}, \text{nfft}, \text{'olap'}, \text{olap}, \dots \text{'order'}, \text{order}, \text{'scale'}, \text{scale}))$

Or add a block on a workbench:



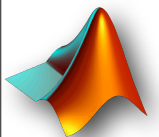
Power Spectral Density Parameters

Name	Description	Values	Default
scale	the output quantity	'PSD' gives Power Spectral Density $[m^2 \text{ Hz}^{-1}]$ 'ASD' gives Amplitude Spectral Density $[m \text{ Hz}^{-1/2}]$ 'PS' gives Power Spectrum $[m^2]$ 'ASD' gives Amplitude Spectrum $[m]$	'PSD'
win	A spectral window to multiply the data by	'BH92' or 'Rectangular' or ... (name or object)	Taken from user preferences
nfft	Window length	-1: one window, length = ao data set Or: length (number of points)	-1
order	Order of segment detrending (prior to windowing)	-1: no detrending 0: mean subtraction N: order N polynomial trend subtraction	0
olap	Percentage overlap between adjacent segments	-1: taken from window parameters 0: no overlap 100: total overlap	-1

Power Spectral Density Estimation (5)



- Features:
 - Multiple inputs:
 - $S = \text{psd}(a1, a2, a3, \text{plist}())$
 - Matrix inputs:
 - $S = \text{psd}([a1 \ a2; a3 \ a4], \text{plist}())$

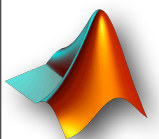


PSD Exercise 1

- Very simple idea:



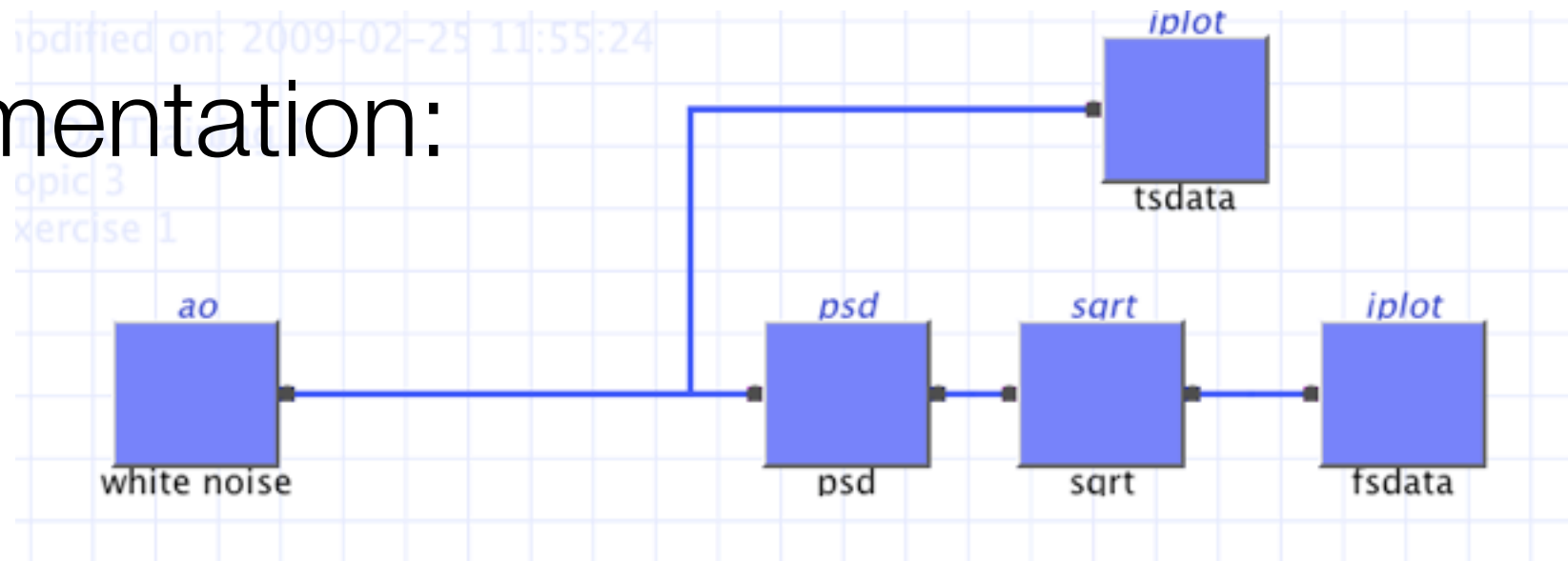
Parameters for psd:
all default



PSD Exercise 1

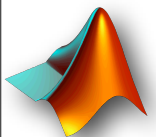


Workbench implementation:



Matlab terminal implementation:

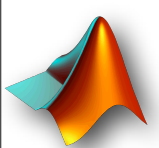
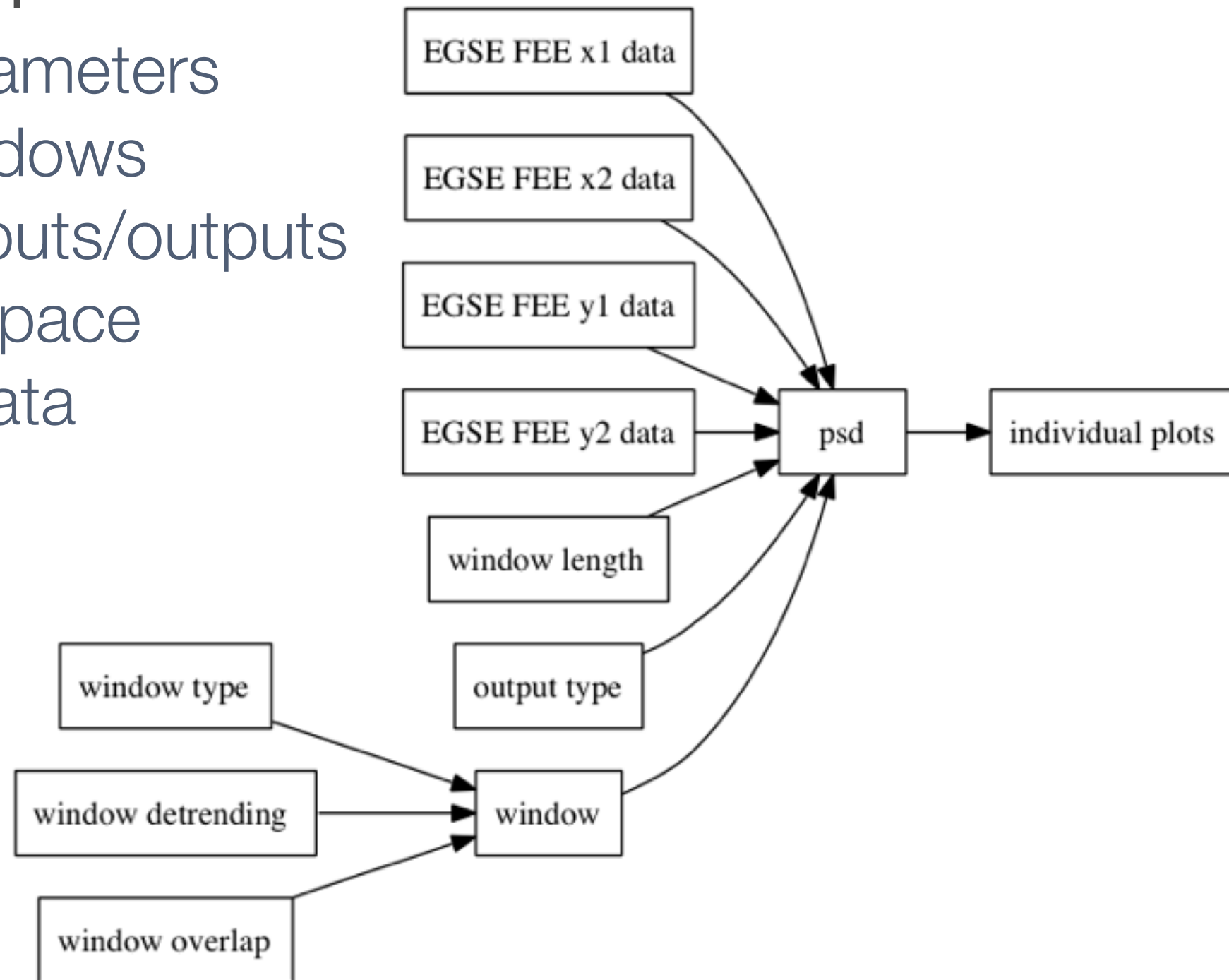
```
>> a1 = ao(plist('waveform','noise','type','normal','nsecs',  
1000,'sigma',1.0,'yunits','m'))  
>> iplot(a1)  
>> S1 = a1.psd(plist('win','BH92'))  
>> P1 = sqrt(S1)  
>> iplot(P1)
```



PSD Exercise 2



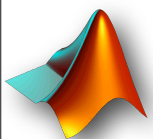
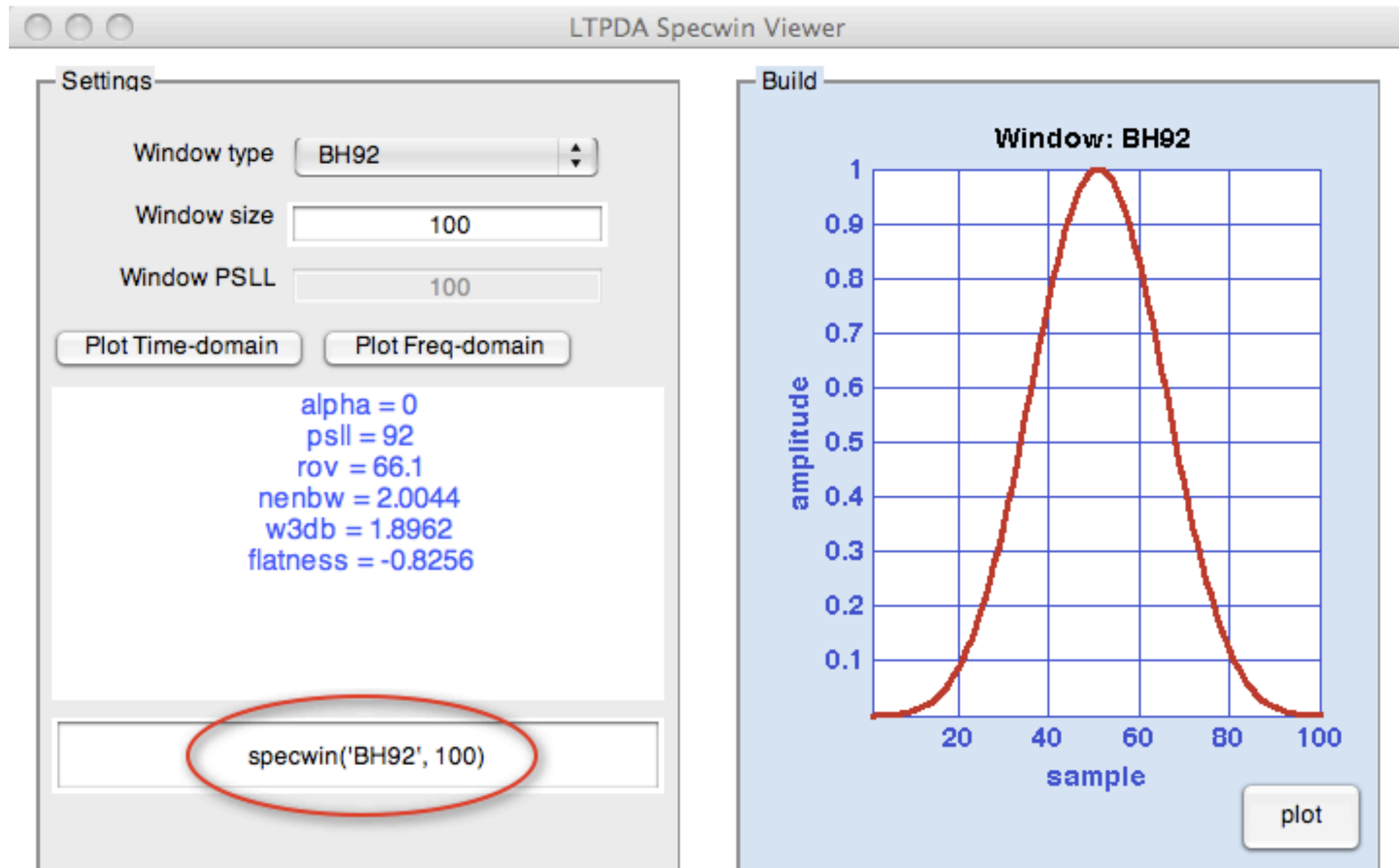
- More involved ...
 - Playing with parameters
 - Playing with windows
 - Adding block inputs/outputs
 - Output to workspace
 - Saving output data



PSD Exercise 2

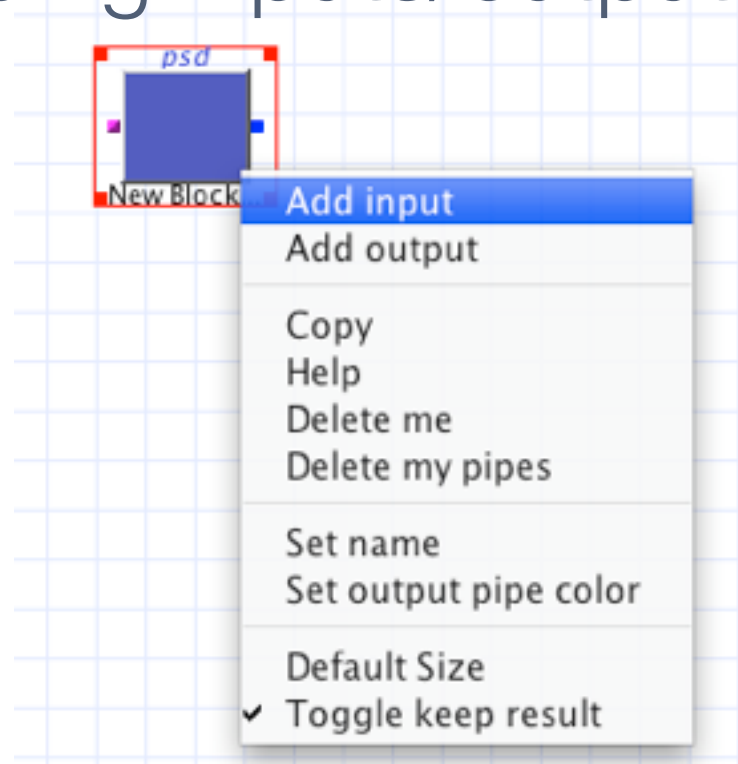


- Spectral windows



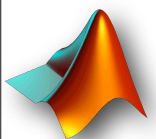
PSD Exercise 2

- Adding inputs/outputs to blocks

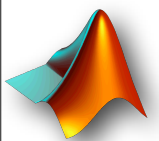
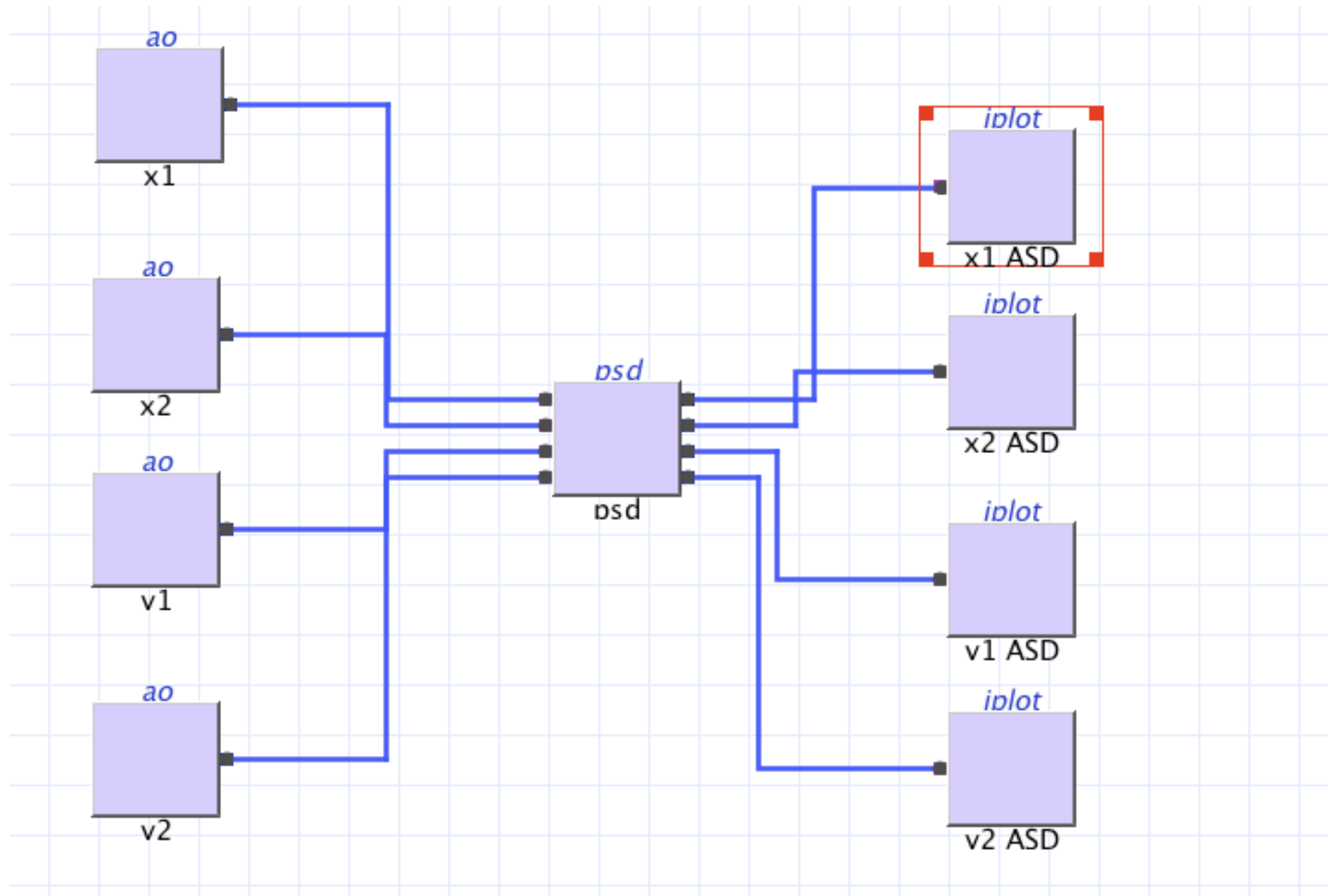


- Playing with parameters

	Key	Value	Edit
<input type="checkbox"/>	KDES	100.0	...
<input type="checkbox"/>	JDES	1000.0	...
<input type="checkbox"/>	LMIN	0.0	...
<input type="checkbox"/>	WIN	BH92	...
<input type="checkbox"/>	PSLL	200.0	...
<input type="checkbox"/>	OLAP	-1.0	...
<input type="checkbox"/>	ORDER	0.0	...
<input type="checkbox"/>	TIMES	[]	...
<input checked="" type="checkbox"/>	SCALE	PSD	...



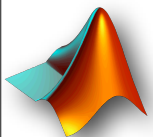
PSD Exercise 2



Log-Scale Power Spectral Density Estimation



- Implementation of the algorithm described in
 - “Measurement 39 (2006) 120-129”
- The same as psd but:
 - Reduces individual point variance by adjusting the window length at each frequency
 - Frequency bins and number of averages are calculated automatically
 - Slower because of requires use of DFT rather than FFT
 - Energy content of the spectrum is preserved
 - Reduced resolution at high frequencies due to shorter window length
 - Lower uncertainty

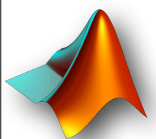


Log-Scale Power Spectral Density Parameters



- Parameters:

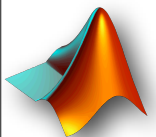
Name	Description	Values	Default
Kdes	Desired number of averages	An integer number	100
Jdes	Desired number of spectral frequencies to calculate	An integer number	1000
Lmin	Minimum segment length	An integer number	0



Log-scale Power Spectral Density Estimation

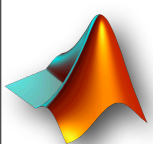
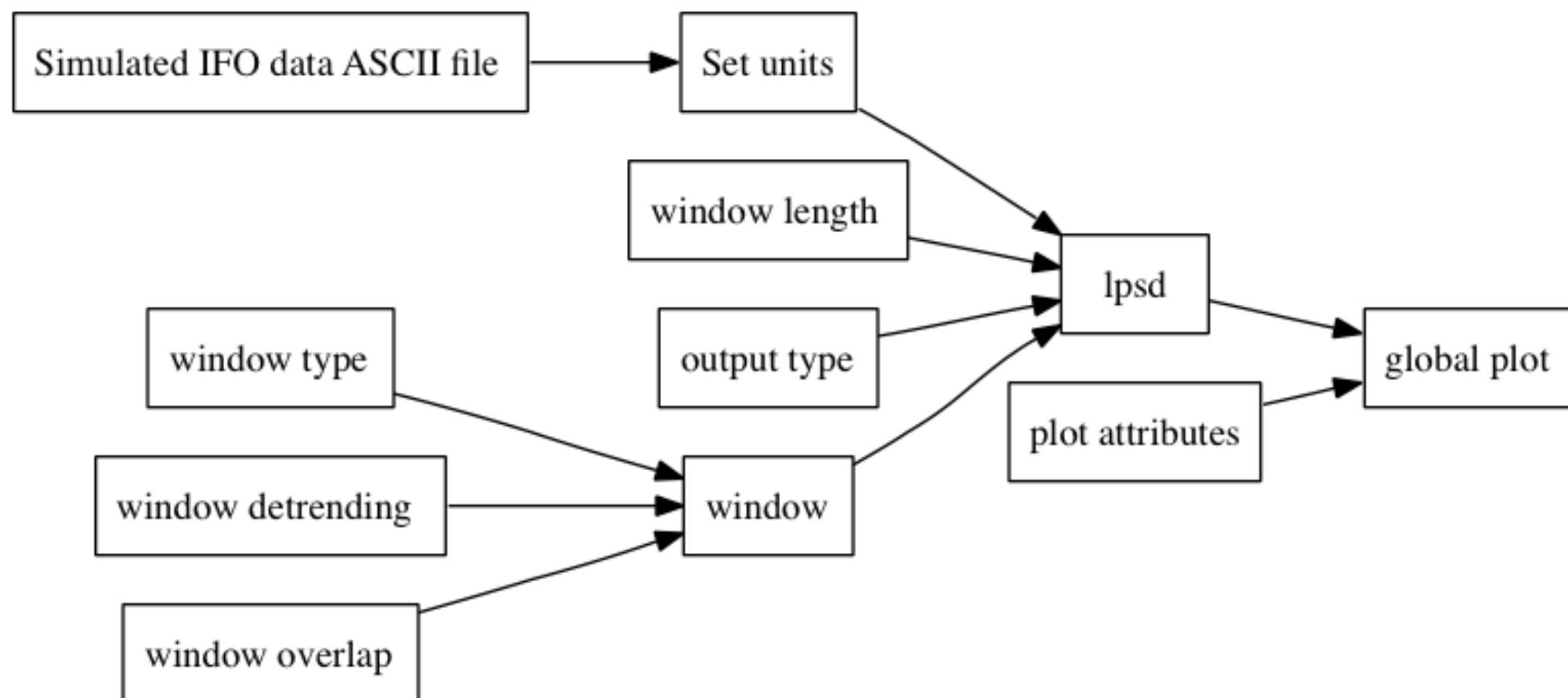


- Features:
- Multiple inputs:
 - $S = \text{Ipsd}(a1, a2, a3, \text{plist}())$
- Matrix inputs:
 - $S = \text{Ipsd}([a1 \ a2; a3 \ a4], \text{plist}())$



PSD Exercise 3

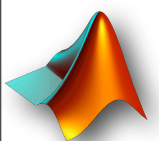
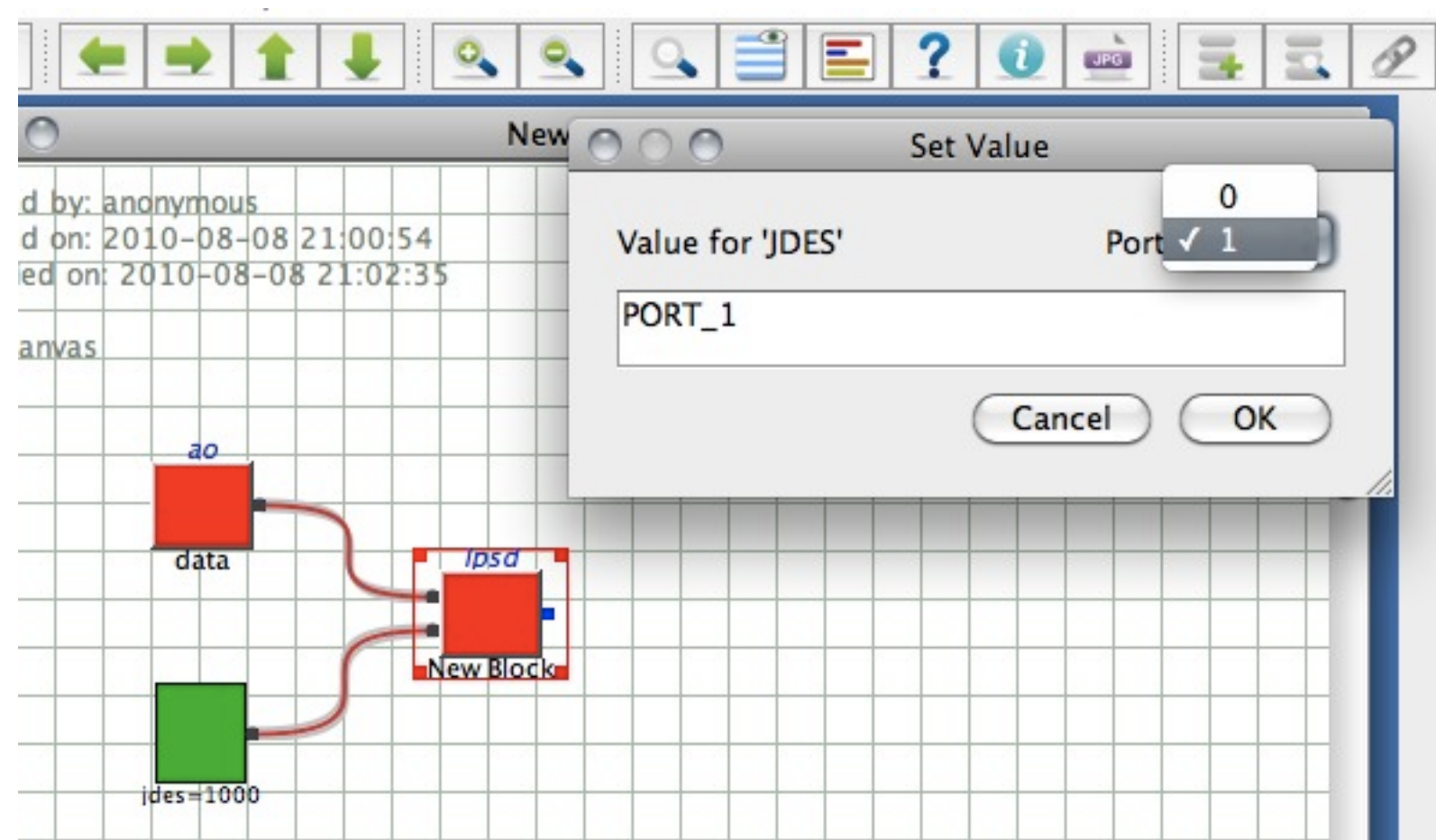
- Using lpsd
 - Log-scale psd calculation to reduce variance
 - Using MDC1 IFO data
 - Setting units
 - Setting plot features



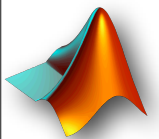
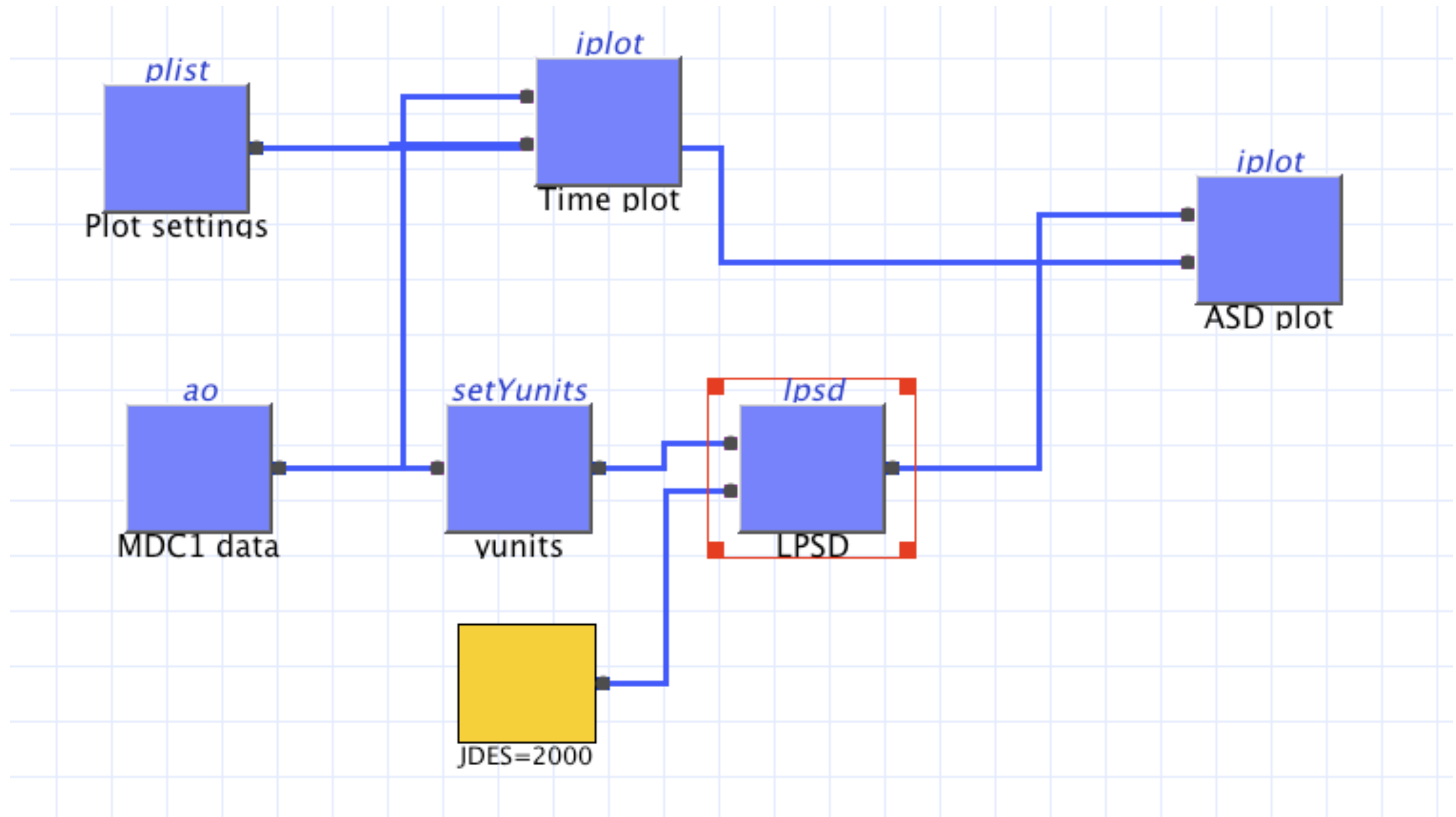
PSD Exercise 3



- Passing values as parameter values



PSD Exercise 3



Cross-Power Spectral Density Estimation

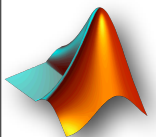


Definition:

$$C_{xy}(f) = \frac{1}{f_s} \sum_{m=-\infty}^{+\infty} R_{xy}(m) \exp(-2\pi i \cdot f \cdot m / f_s)$$

Estimates the *one-sided* CPSD:

$$\tilde{C}_{xy}(f) = \begin{cases} 0, & -\frac{f_s}{2} \leq f \leq 0 \\ 2P_{xy}(f), & 0 \leq f \leq \frac{f_s}{2} \end{cases}$$



Cross- Power Spectral Density Estimation (2)

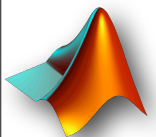


- Use Welch method as in PSD
- The CPSD at each frequency f is estimated as:

$$\hat{C}_{xy}(f) = \frac{X_L Y_L^*}{f_s \cdot L \cdot U}$$

- where:

$$x_L(f) = \sum_{n=0}^{L-1} x_L[n] \exp\left(-2\pi i \cdot \frac{f \cdot n}{f_s}\right) \quad U = \frac{1}{L} \sum_{n=0}^{L-1} |w(n)|^2$$





- Methods:
 - ao/cpsd: linear frequency scale
 - ao/lcpsd: log-frequency scale

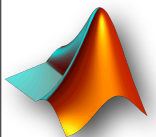
Similarly, we can evaluate coherence:

$$Coh_{xy}(f) = \frac{|C_{xy}(f)|^2}{P_{xx}(f)P_{yy}(f)}$$

Methods:

ao/cohere: linear frequency scale

ao/lcohere: log-frequency scale

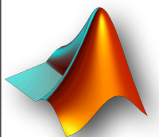


Cross- Power Spectral Density Estimation (4)



```
p1 = plist(...  
    'win',win,...  
    'nfft',nfft,...  
    'olap',olap,...  
    'order',order,...  
    'scale',scale);
```

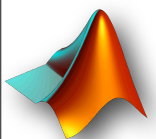
```
Cxy = cpsd(x, y, p1)
```



Cross- Power Spectral Density Parameters



Name	Description	Values	Default
win	A spectral window to multiply the data by	'BH92' or 'Rectangular' or ... name or object	Taken from user preferences
nfft	Length of the window	-1 : one window, length = ao data set length Or: length (number of points)	-1
order	Order of segment detrending (prior to windowing)	-1 : no detrending 0: mean subtraction N: order N polynomial trend subtraction	0
olap	Percentage overlap between adjacent segments	-1 : taken from window parameters 0: no overlap 100: total overlap	-1

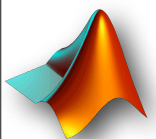


Transfer Function Estimation

- Methods:
 - ao/tfe: linear frequency scale
 - ao/ltfe: log-frequency scale
- Definition:

$$T_{xy}(f) = \frac{C_{xy}(f)}{P_{yy}(f)}$$

- Use Welch method again

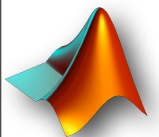


Transfer Function Estimation (2)



```
pl = plist(...  
    'win', win, ...  
    'nfft', nfft, ...  
    'olap', olap, ...  
    'order', order, ...  
    'scale', scale);
```

```
Txy = tfe(x, y, pl)
```

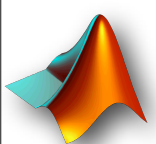


Transfer Function Estimation Parameters



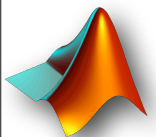
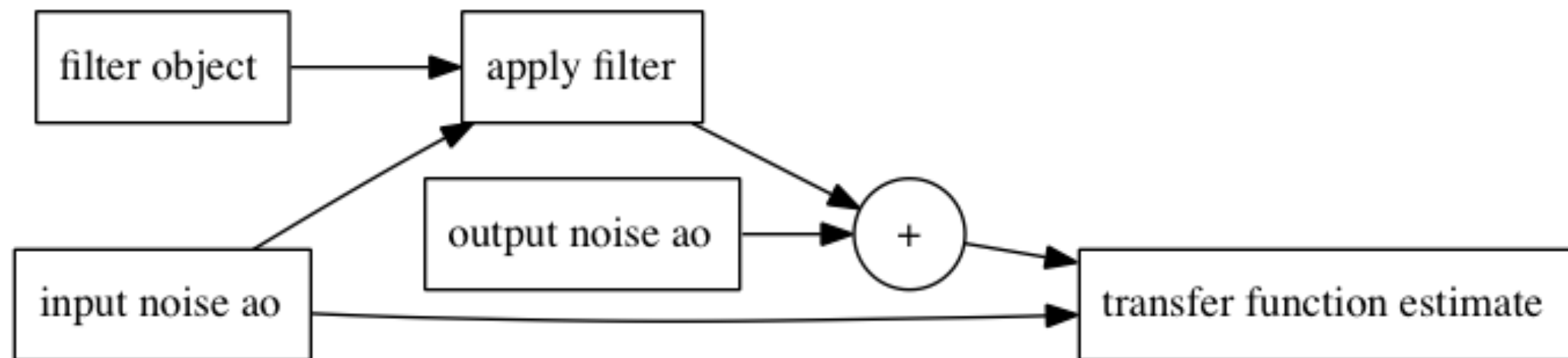
Name	Description	Values	Default
win	A spectral window to multiply the data by	'BH92' or 'Rectangular' or ... name or object	Taken from user preferences
nfft	Length of the window	-1 : one window, length = ao data set length Or: length (number of points)	-1
order	Order of segment detrending (prior to windowing)	-1 : no detrending 0: mean subtraction N: order N polynomial trend subtraction	0
olap	Percentage overlap between adjacent segments	-1 : taken from window parameters 0: no overlap 100: total overlap	-1

And similarly for **ltfe** ...



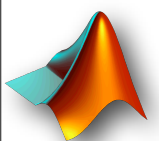
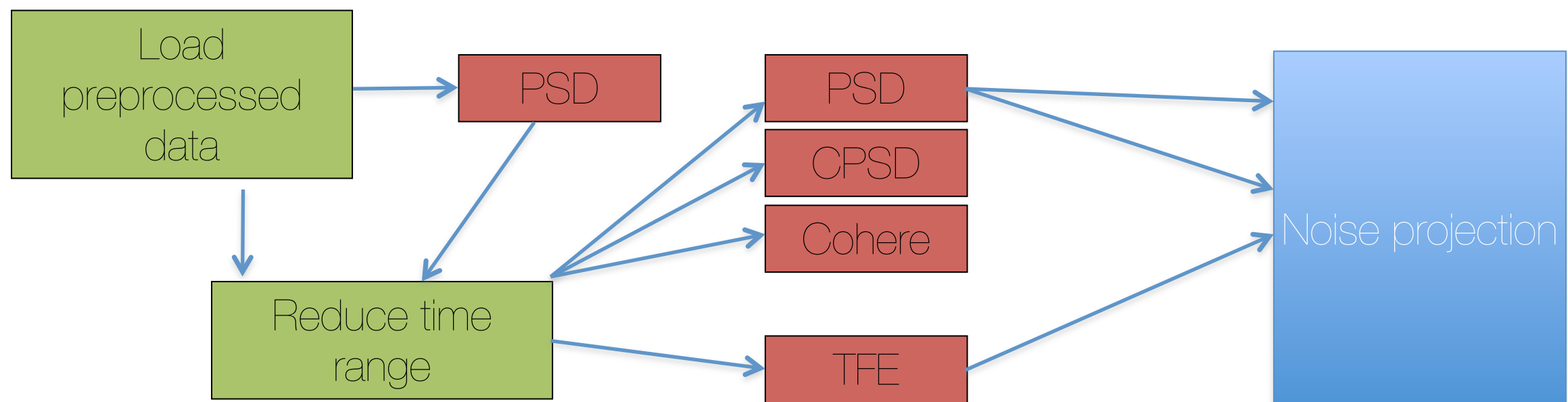
TFE Exercise 1

- Using tfe
 - Simulated data input: white noise
 - Band-pass filter object
 - Filtering the input noise
 - Adding output white noise
 - Estimate the transfer function



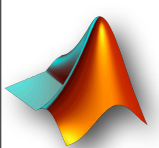
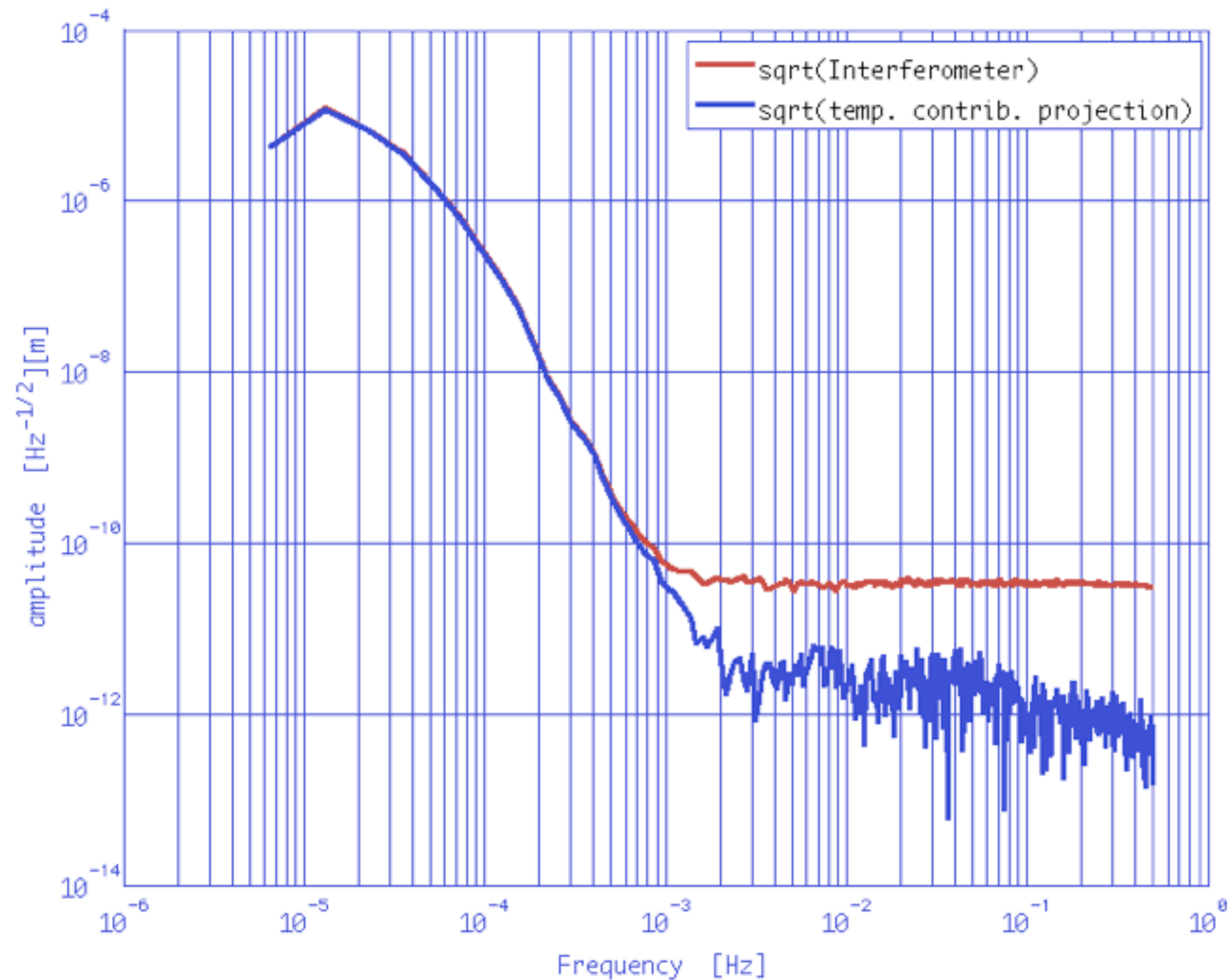
IFO/Temperature Example

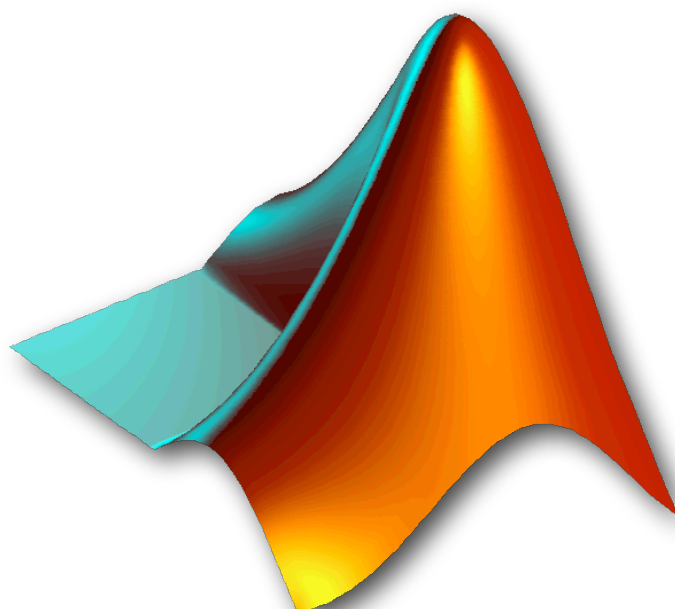
- Estimating the empirical transfer function:
temperature \rightarrow position
 - Load preprocessed data
 - Evaluate PSD of T and x
 - Reduce the time range
 - Evaluate CPSD and cross-coherence of T and x
 - Estimate the transfer function of T into x
 - Perform the noise projection



IFO/Temperature Example

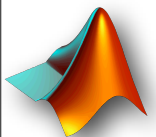
- We are aiming to obtain:





Topic 4: Transfer function models and digital filters

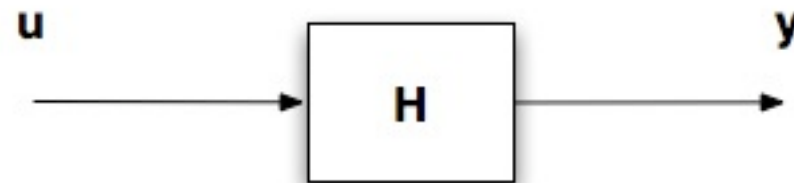
- Transfer function models in s domain
 - Pole zero representation
 - Rational representation
 - Partial fraction representation
- Transformation between representations
- Modeling a system
- Filtering data
 - discretizing a model
 - setting filter properties
- IFO/Temperature example



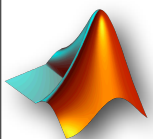
Overview



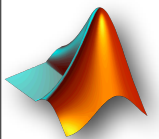
- The general scheme: input, output and a transfer function



- Aim of this topic:
 - How to model the transfer function H in continuous domain, $H = H(s)$
 - How to discretize our model $H(s) \rightarrow H(z)$
 - How to filter data with $H(z)$
 - How to define $H(z)$ from filter properties



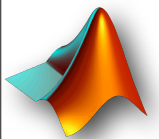
Tools used here



Tools used here

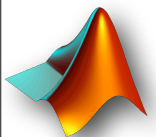
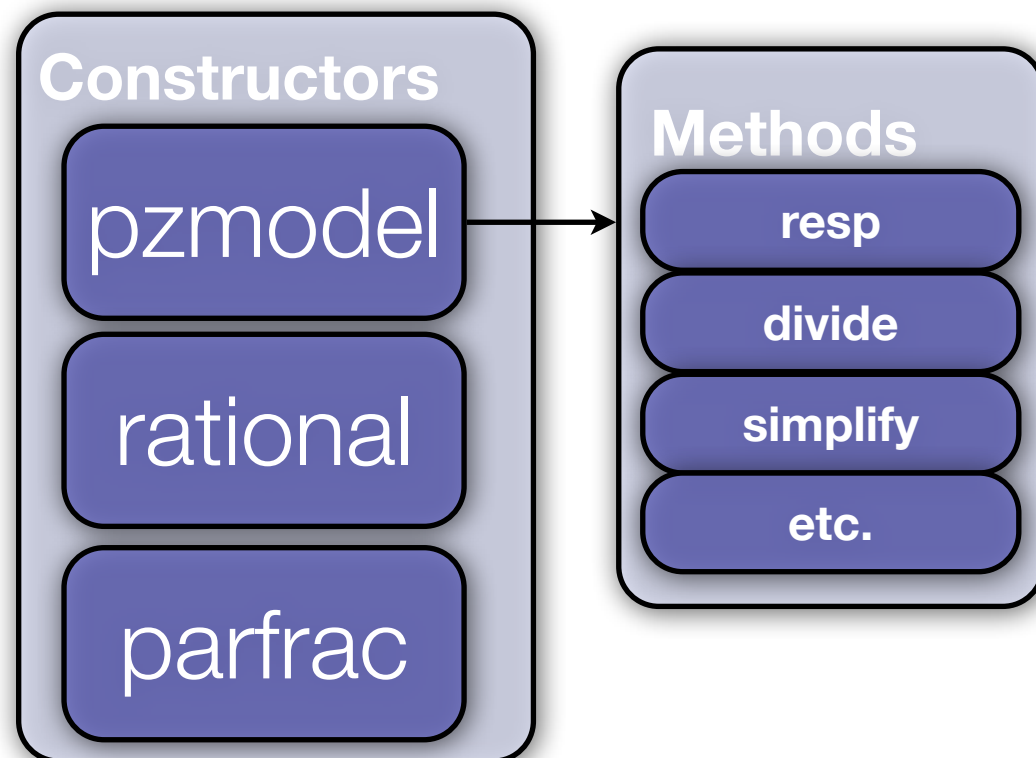


1. Continuous domain



Tools used here

1. Continuous domain

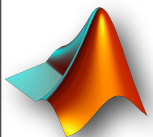
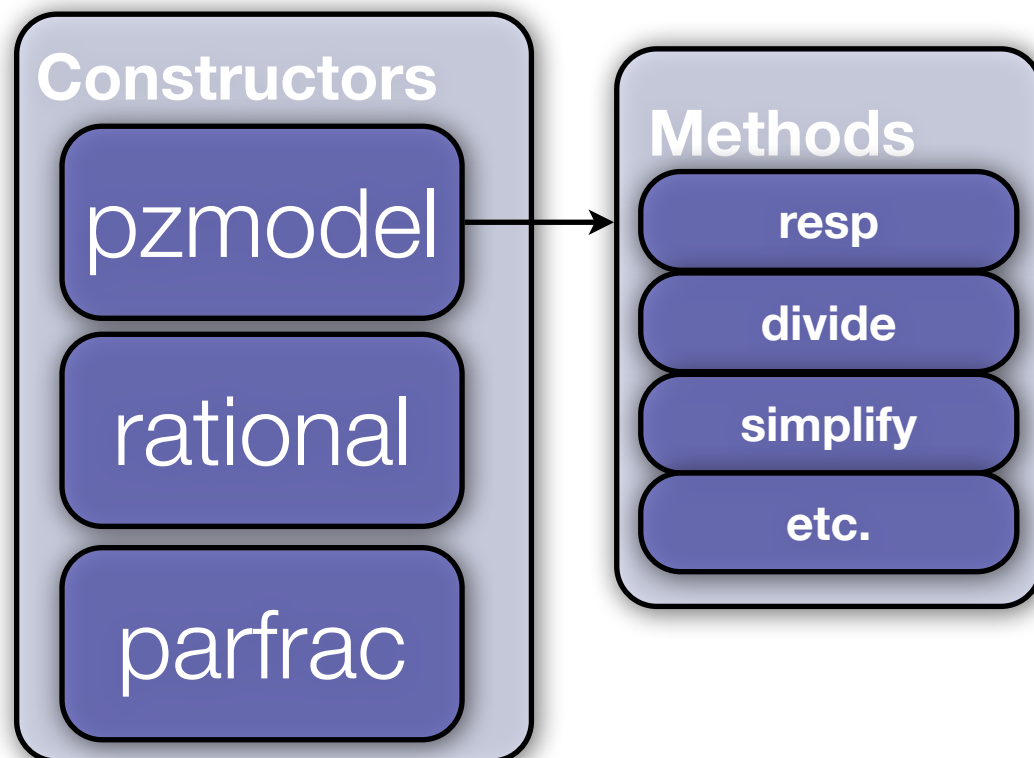


Tools used here



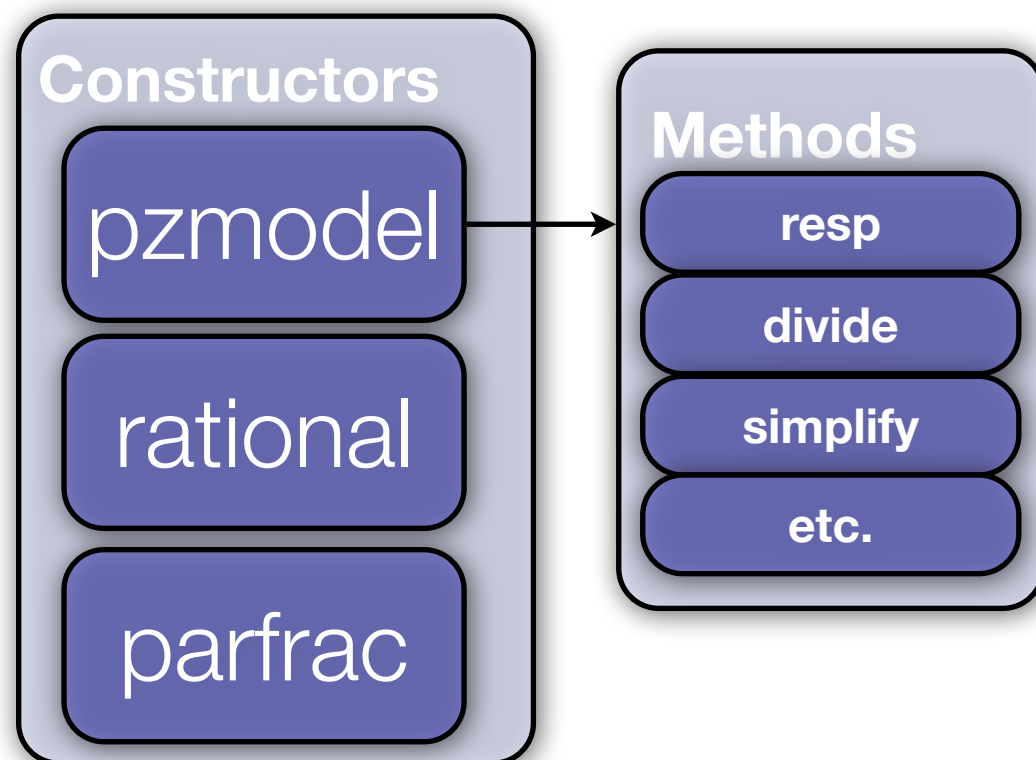
1. Continuous domain

2. Discrete domain

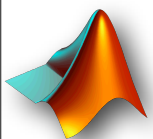
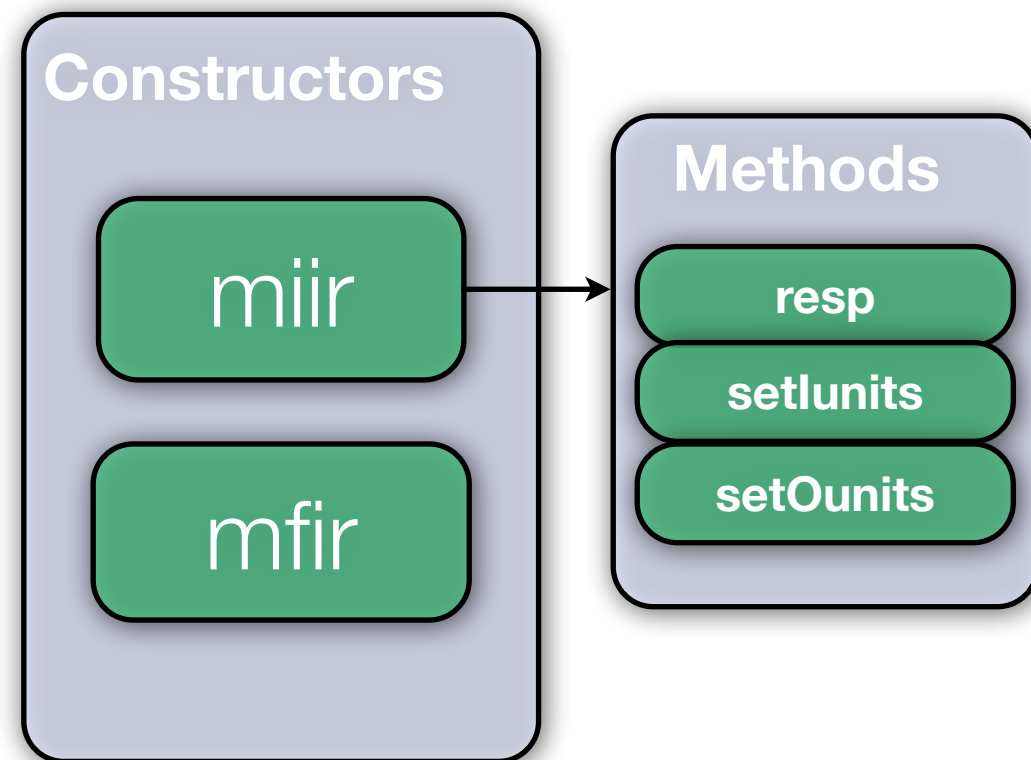


Tools used here

1. Continuous domain

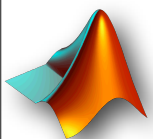
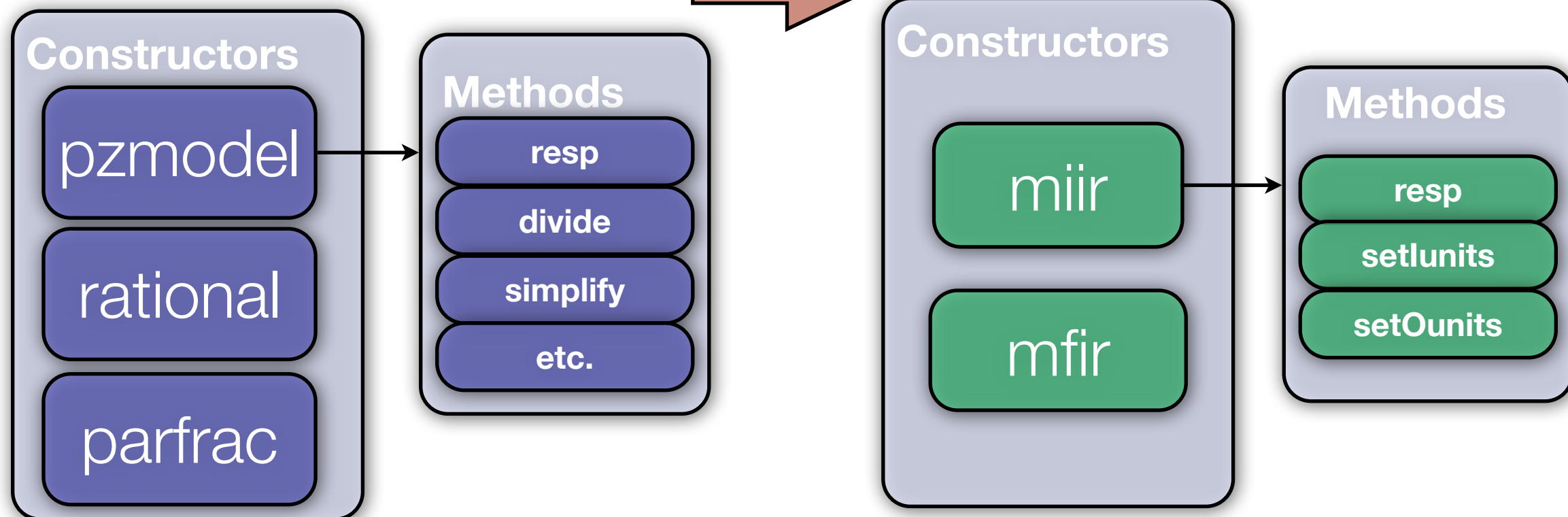


2. Discrete domain

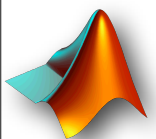
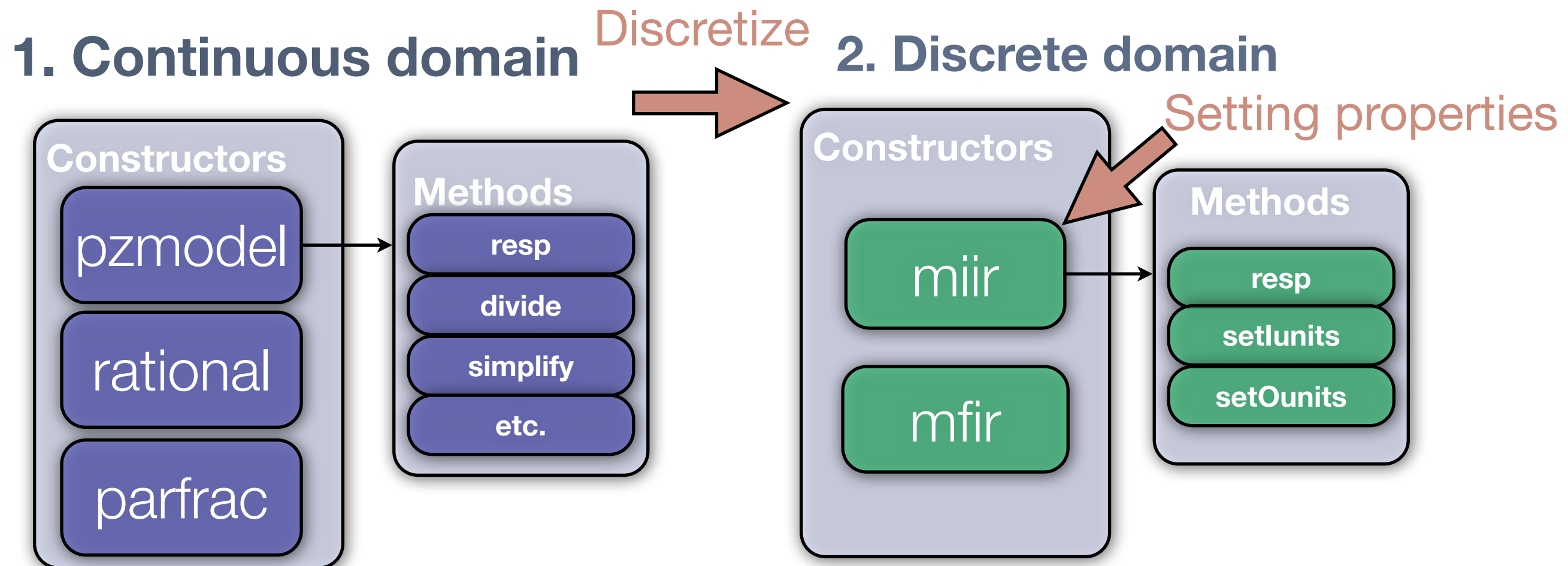


Tools used here

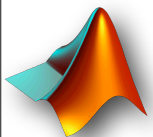
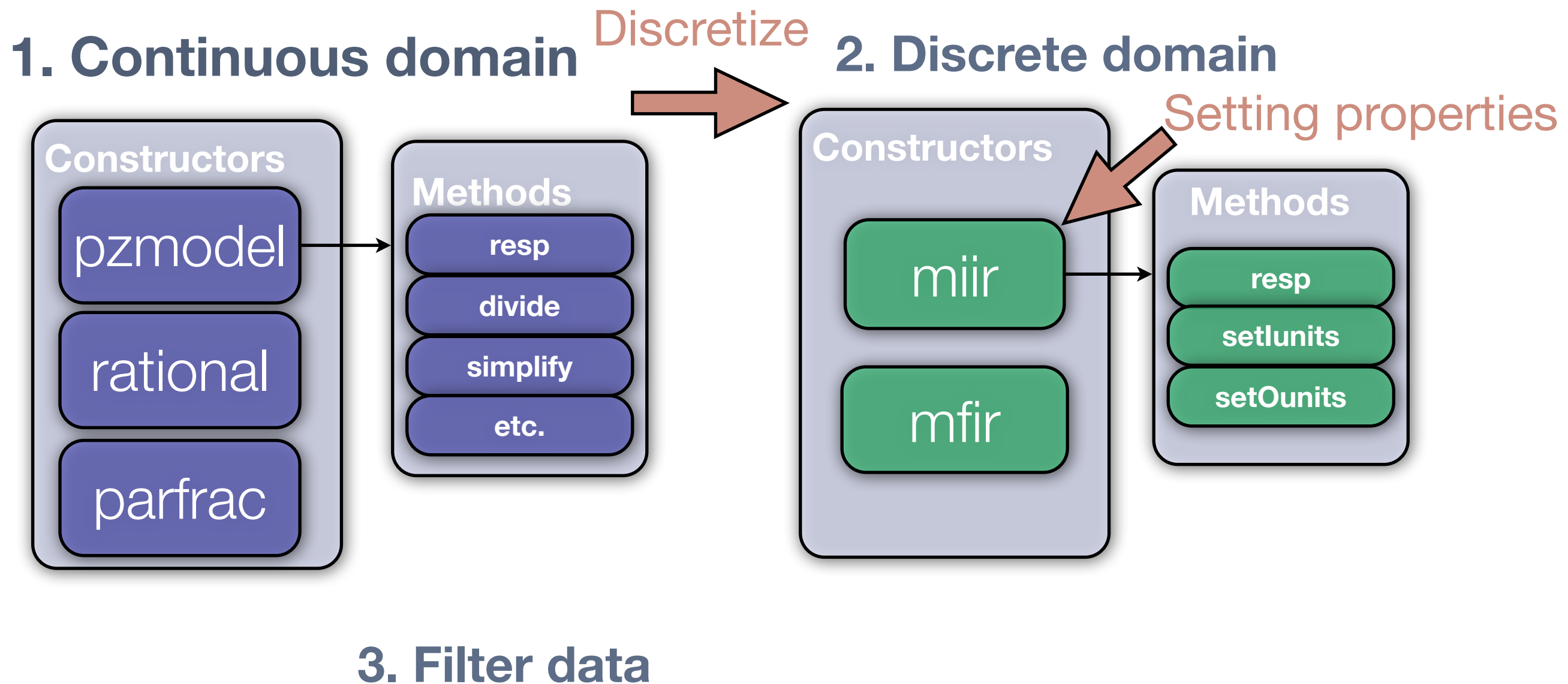
1. Continuous domain Discretize 2. Discrete domain



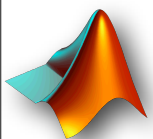
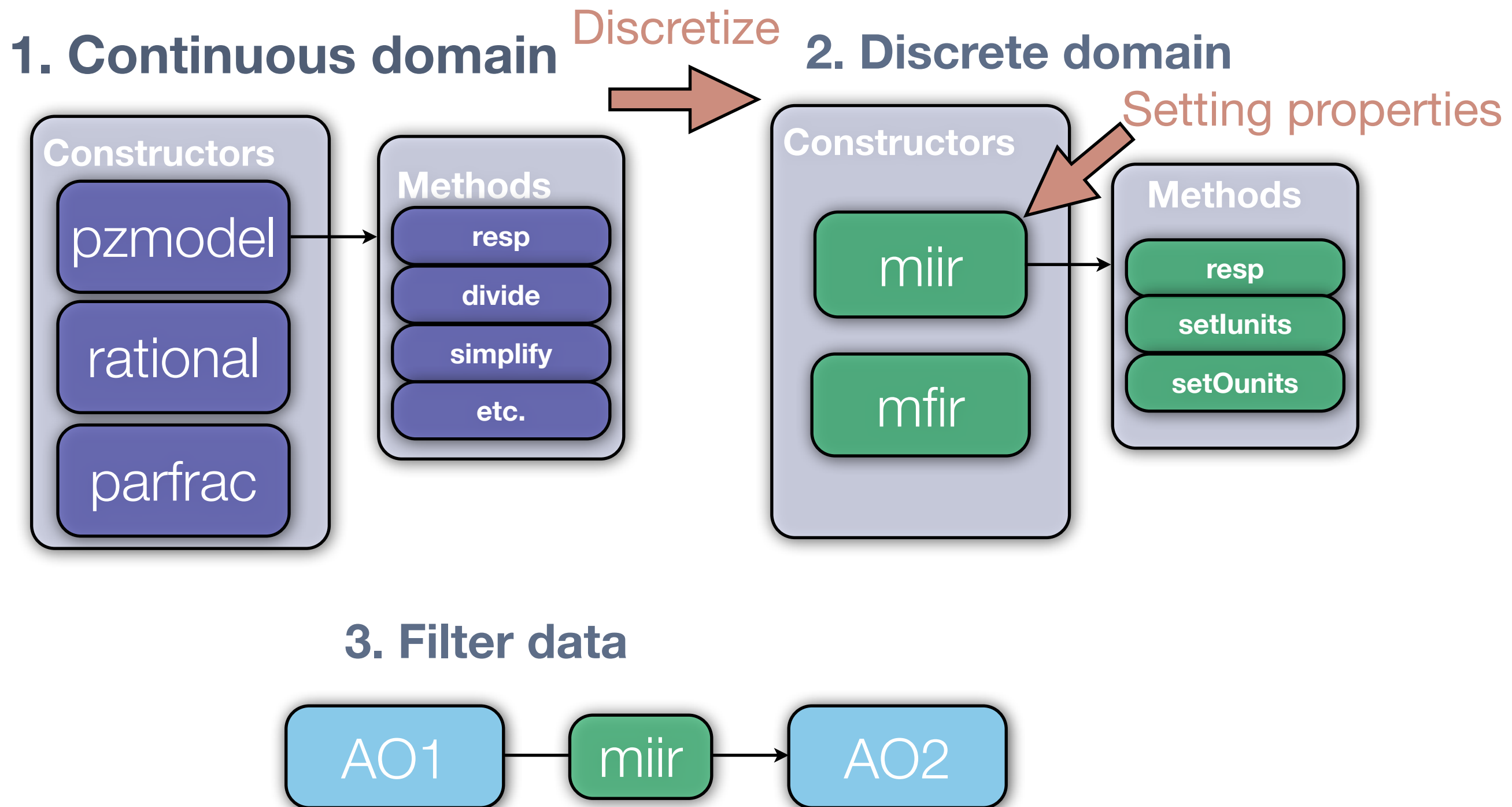
Tools used here



Tools used here



Tools used here

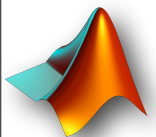


Pole/zero models

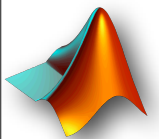
- A pole zero model is defined by:
 - Gain, poles, zeros, delay

$$H(s) = G \frac{(s - z_1)(s - z_2) \dots (s - z_n)}{(s - p_1)(s - p_2) \dots (s - p_m)} e^{-i\omega\tau}$$

- LTPDA constructor: PZMODEL
 - PZMODELS can be multiplied and divided
 - Delay is added or subtracted in such a case



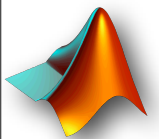
About poles (and zeros) notation



About poles (and zeros) notation

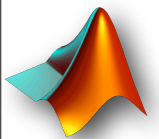
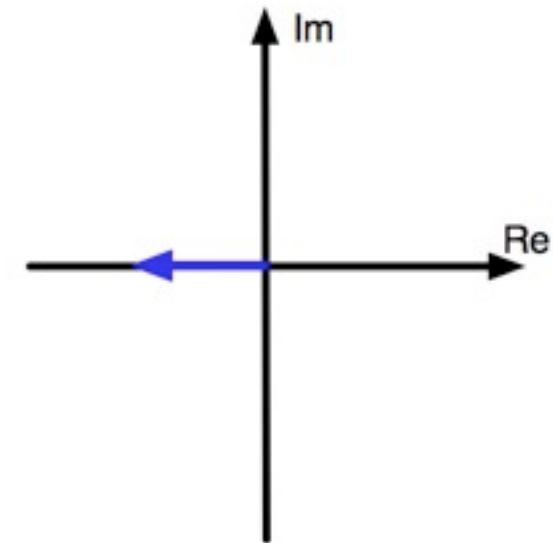


- Simple pole: $f = 1 \text{ Hz}$



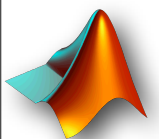
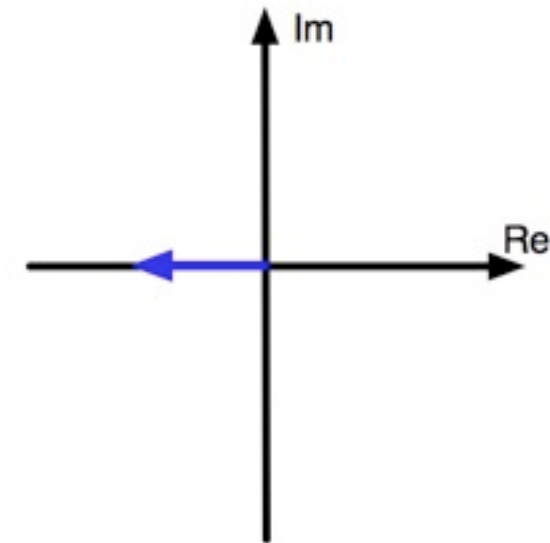
About poles (and zeros) notation

- Simple pole: $f = 1 \text{ Hz}$



About poles (and zeros) notation

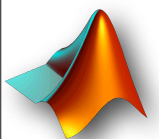
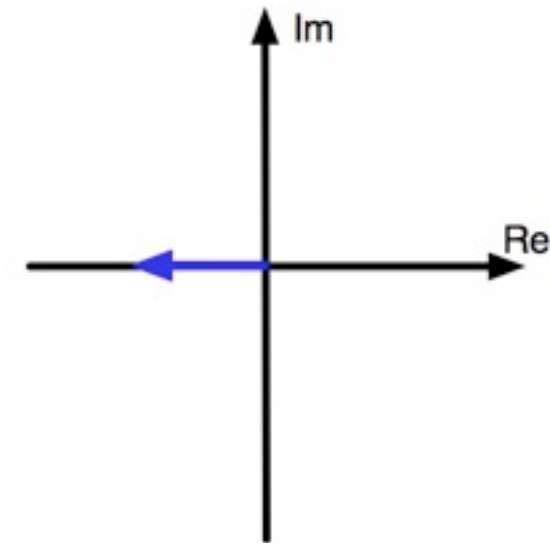
- Simple pole: $f = 1 \text{ Hz}$
- Pole pairs: $(f = 1 \text{ Hz}, Q)$



About poles (and zeros) notation

- Simple pole: $f = 1 \text{ Hz}$
- Pole pairs: $(f = 1 \text{ Hz}, Q)$

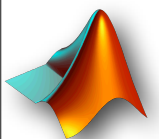
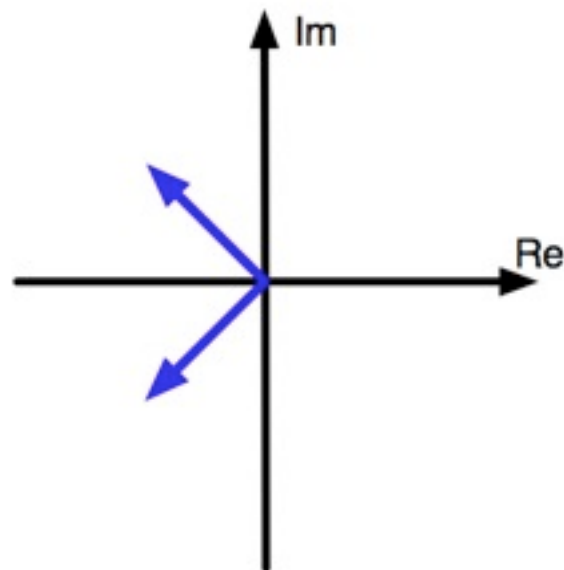
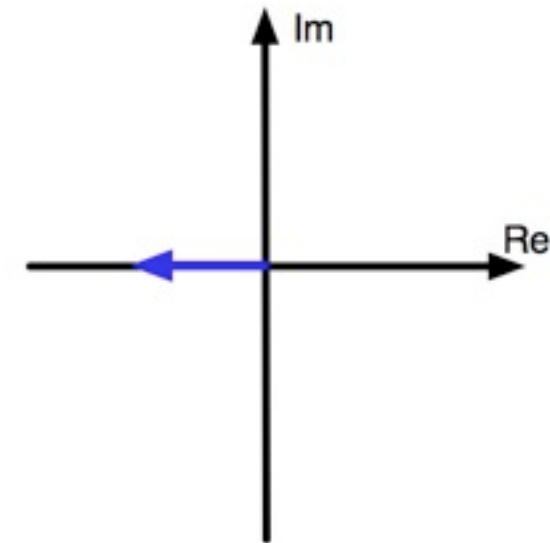
$Q > 0.5$
(underdamped)



About poles (and zeros) notation

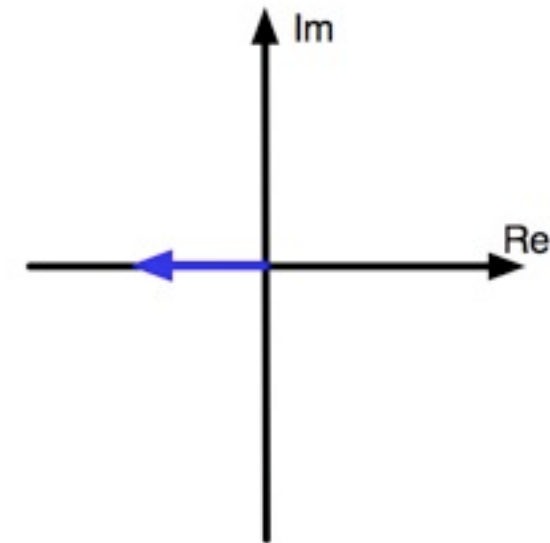
- Simple pole: $f = 1 \text{ Hz}$
- Pole pairs: ($f = 1 \text{ Hz}$, Q)

$Q > 0.5$
(underdamped)



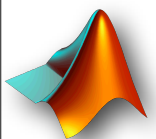
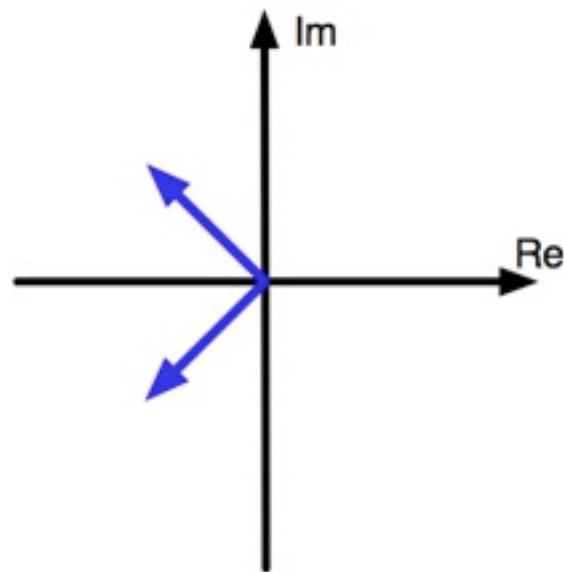
About poles (and zeros) notation

- Simple pole: $f = 1$ Hz
- Pole pairs: ($f = 1$ Hz, Q)



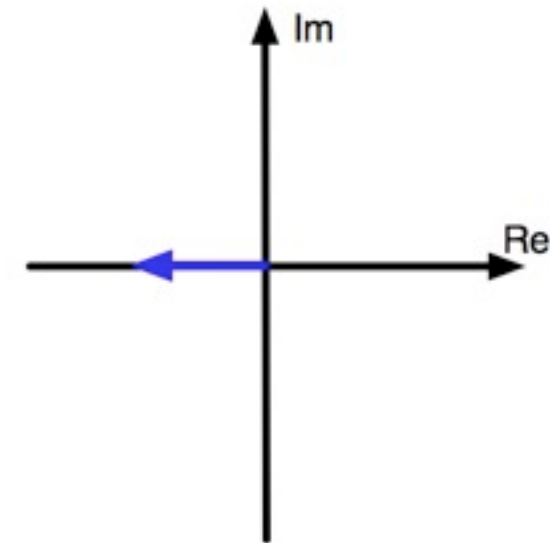
$Q > 0.5$
(underdamped)

$Q = 0.5$
(critically damped)

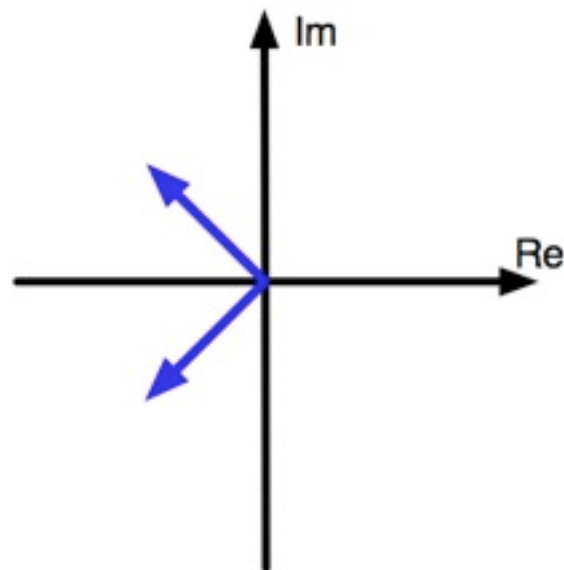


About poles (and zeros) notation

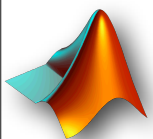
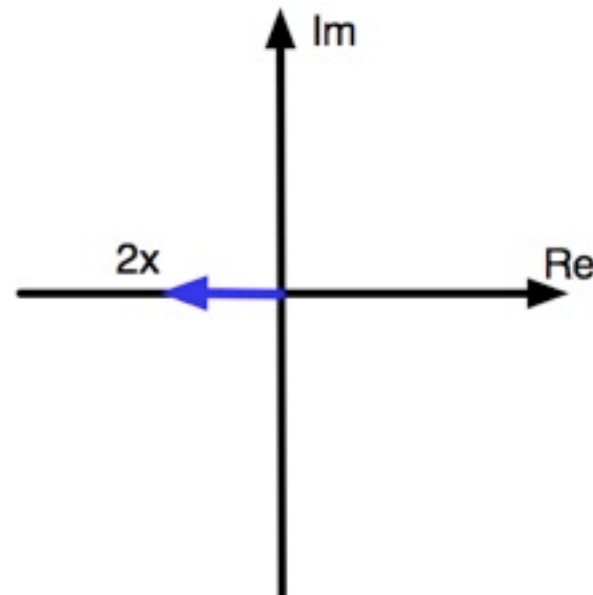
- Simple pole: $f = 1$ Hz
- Pole pairs: ($f = 1$ Hz, Q)



$Q > 0.5$
(underdamped)

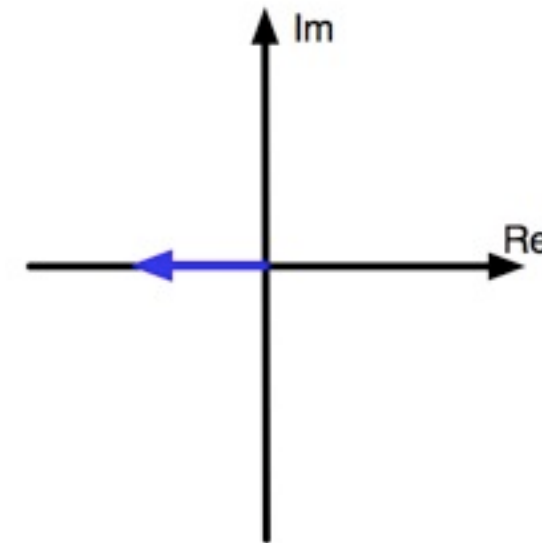


$Q = 0.5$
(critically damped)

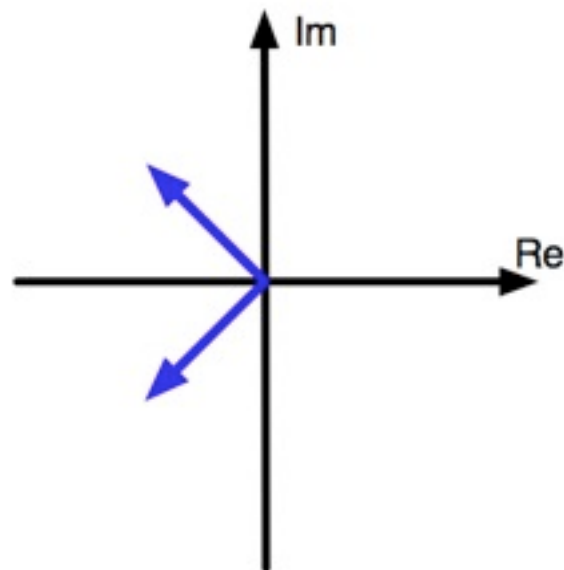


About poles (and zeros) notation

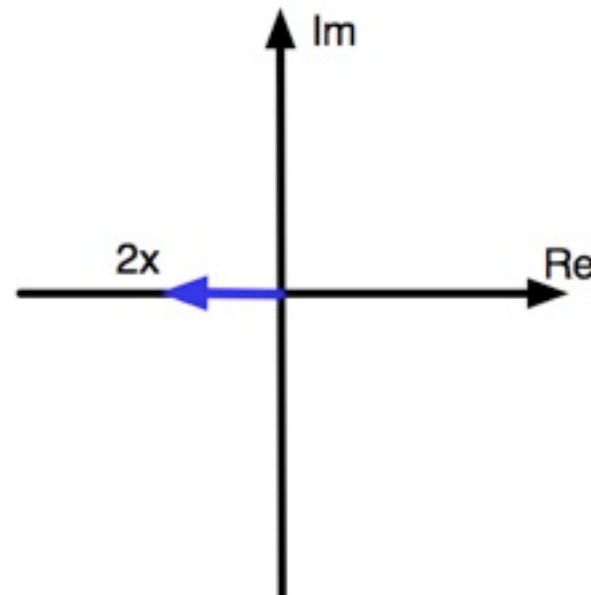
- Simple pole: $f = 1$ Hz
- Pole pairs: ($f = 1$ Hz, Q)



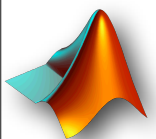
$Q > 0.5$
(underdamped)



$Q = 0.5$
(critically damped)

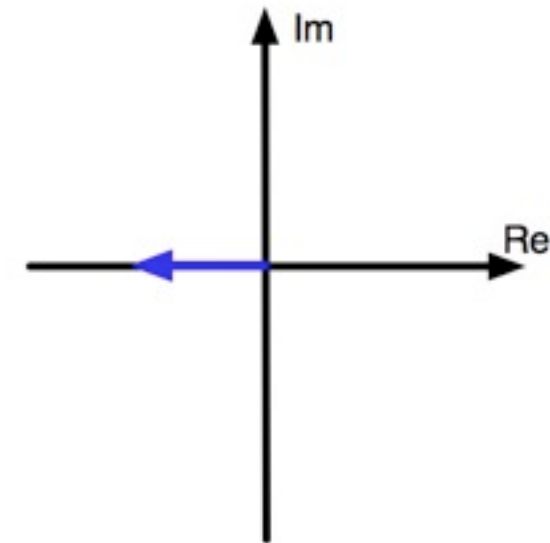


$Q < 0.5$
(overdamped)

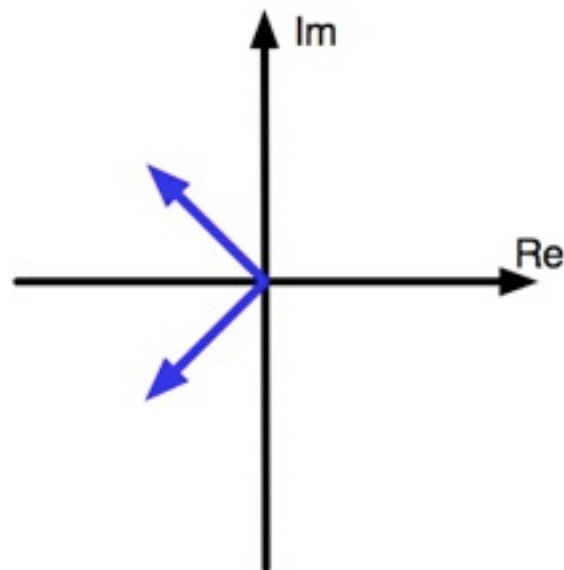


About poles (and zeros) notation

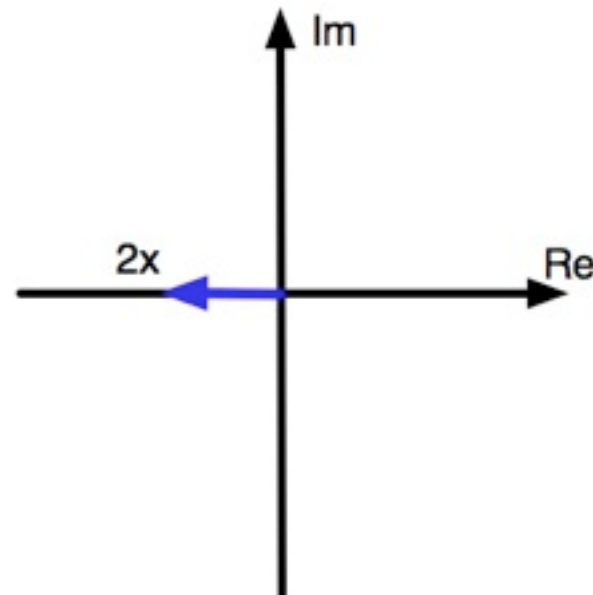
- Simple pole: $f = 1$ Hz
- Pole pairs: ($f = 1$ Hz, Q)



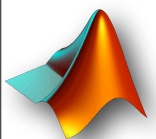
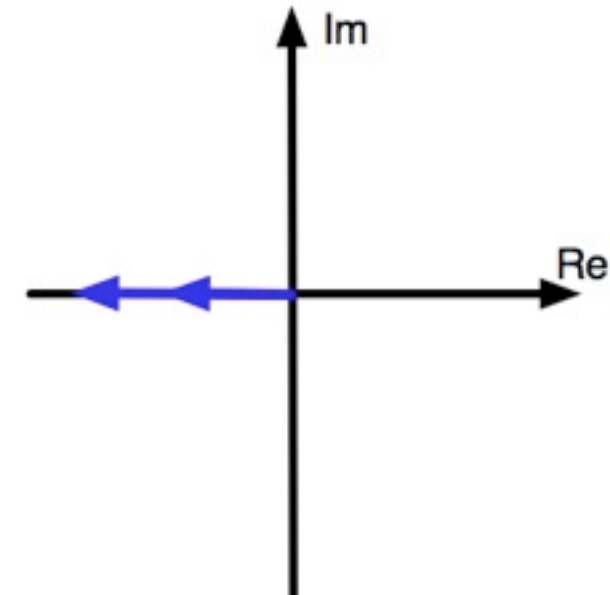
$Q > 0.5$
(underdamped)



$Q = 0.5$
(critically damped)



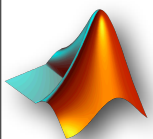
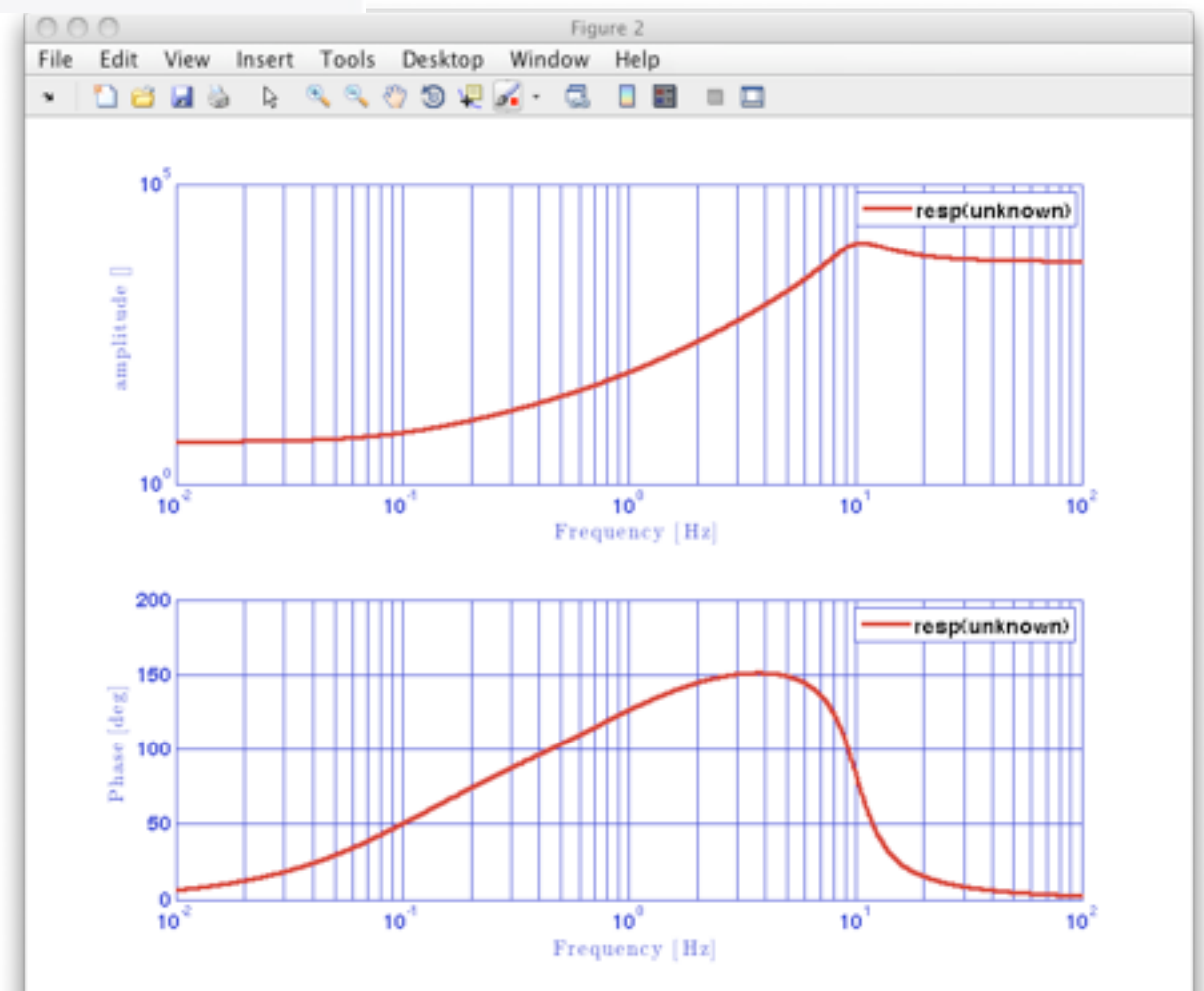
$Q < 0.5$
(overdamped)



Pole/zero models

- Working example: Compute pole zero response
 - Topic 4 -> Create transfer function -> Create pole zero model

Key	Value
GAIN	5
POLES	(f = 1 Hz, Q = 2)
ZEROS	(f = 1 Hz), (f = 0.1 Hz)

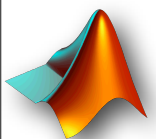


Rational models

- A rational model is defined by:
 - Num. and den. coefficients

$$H(s) = \frac{a_1 s^m + a_2 s^{m-1} + \dots + a_{m+1}}{b_1 s^n + b_2 s^{n-1} + \dots + b_{n+1}}$$

- LTPDA constructor: RATIONAL
 - RATIONALs can NOT be multiplied and divided
- Working example: Compute rational response
 - Topic 4 -> Create transfer func... -> Create rational model

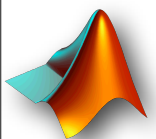


Partial-fraction models

- A partial fraction model is defined by:
 - Poles, residues and direct terms

$$H(s) = K(s) + \sum_{i=1}^N \frac{R_i}{s - p_i}$$

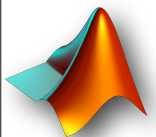
- LTPDA constructor: PARFRAC
- PARFRACs can NOT be multiplied and divided
- Working example: Compute par. frac. response
 - Topic 4 -> Create transfer func... -> Create par. frac. model



Transforming models

- Some of the possible transformations are implemented in v2.3
 - Works by inputting an object into a constructor
 - e.g. `rat = rational(pzm)`
- Working example: `pzmodel -> rational -> pzmodel`
 - Topic 4 > Transforming models between representations

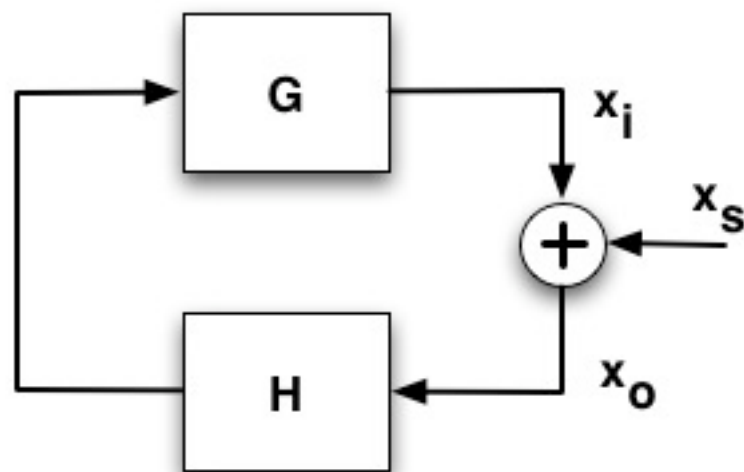
	Pole/Zero	Rational	Partial Fraction
Pole/Zero		✓	✓
Rational	✓		✓
Partial Fraction	✓	✓	



Modeling a system

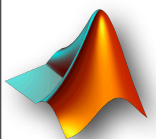
- Pole zero model
 - Modelling a closed loop system with pzmodel
 - Basic pzmodel operations

Our system:



$$\frac{x_o}{x_s} = \text{CLG} = \frac{1}{1 - \text{OLG}} = \frac{1}{1 - H \cdot G}$$

- The problem:
 - Assuming OLG and H known, determine G and CLG



Modeling a system

- Step-by-step

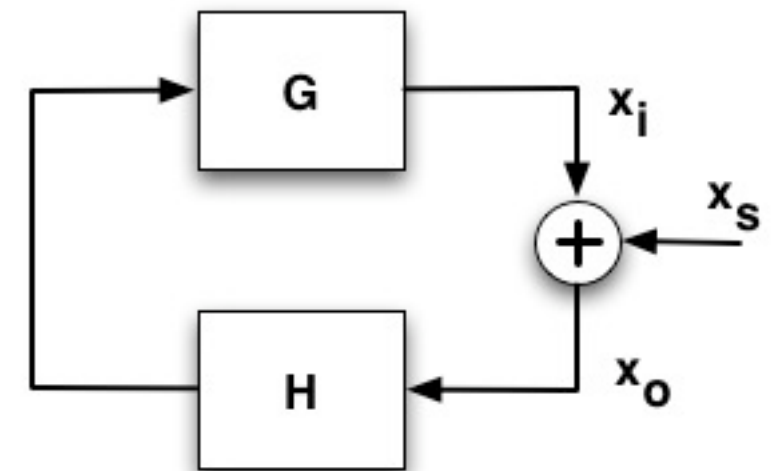
1. $G = \text{OLG}/H$

(*G is a pzmodel*)

2. Operate on G: setName, simplify ...

3. $\text{CLG} = 1/(1-\text{OLG})$

(*CLG is NOT a pzmodel*)

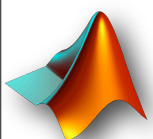


$$\frac{x_o}{x_s} = \text{CLG} = \frac{1}{1 - \text{OLG}} = \frac{1}{1 - H \cdot G}$$

- Repeat loading H with delay

- Working example: Modeling a system

- Topic 4 > Modeling a system



Entering the discrete domain

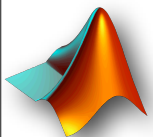
- The LTPDA toolbox allows you to build digital filters...
 - Discretizing your model
 - Example: find the filters for H,G, OLG in our closed loop
- Defining filter properties
 - Example: Design a bandpass filter to evaluate power spectrum in a bandwidth
- Filter constructors in LTPDA

- MIIR (IIR filters)

$$y[n] = \sum_{k=0}^N b[k] x[n-k] - \sum_{k=1}^M a[k] y[n-k]$$

- MFIR (FIR filters)

$$y[n] = \sum_{k=0}^M b[k] x[n-k]$$



By discretizing a transfer function

- Syntax: insert pzmodel into constructor

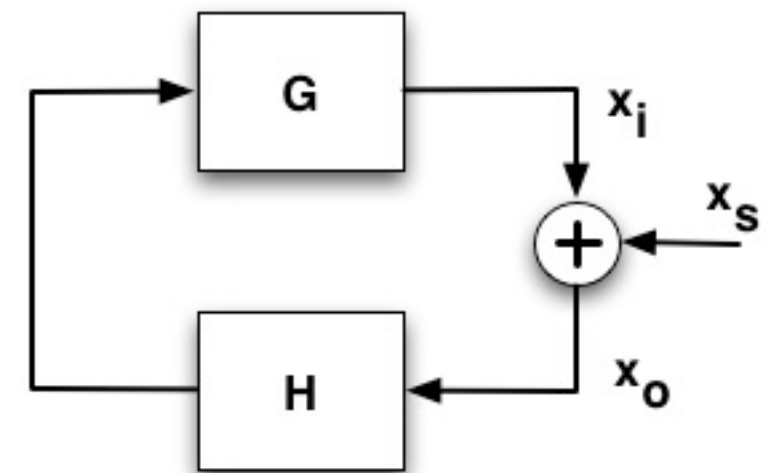
```
Gd = miir(G,plist('fs',10));
```

Constructor
 pzmodel obj.
 filter obj.

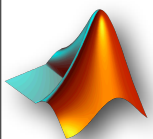
- Step-by-step

1. Discretize G,H,OLG
2. Compare continuous and digital response
3. Get filter coefficients

- Delay is NOT used in the discretization!



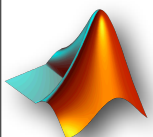
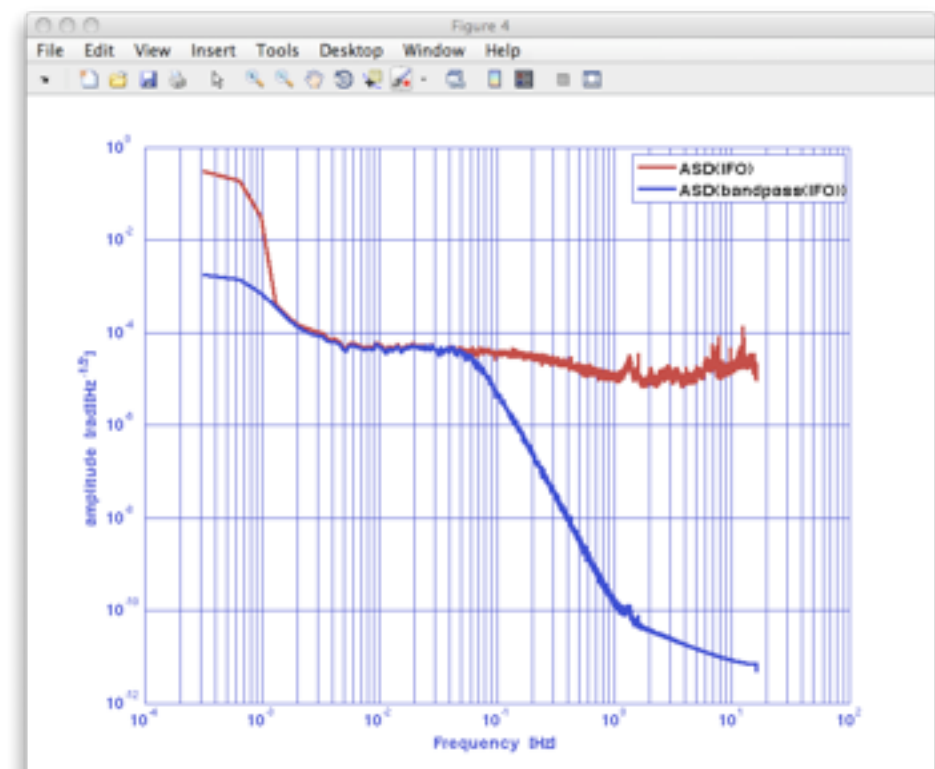
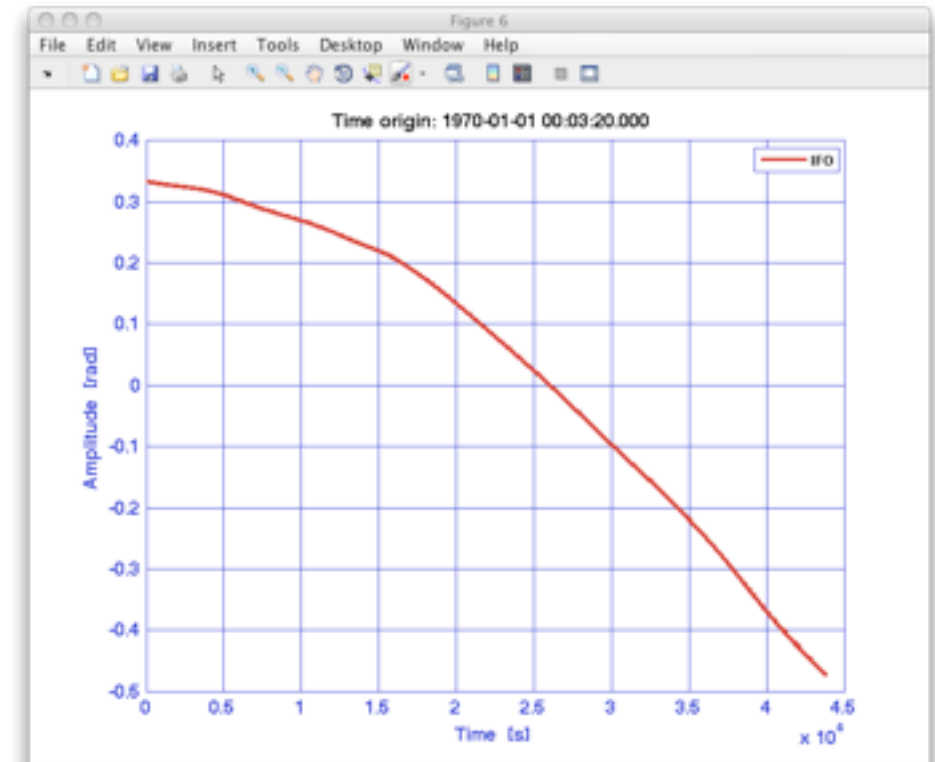
- Working example: Get filters for closed loop pzmodels
 - Topic 4 > How to filter data > By discretizing...



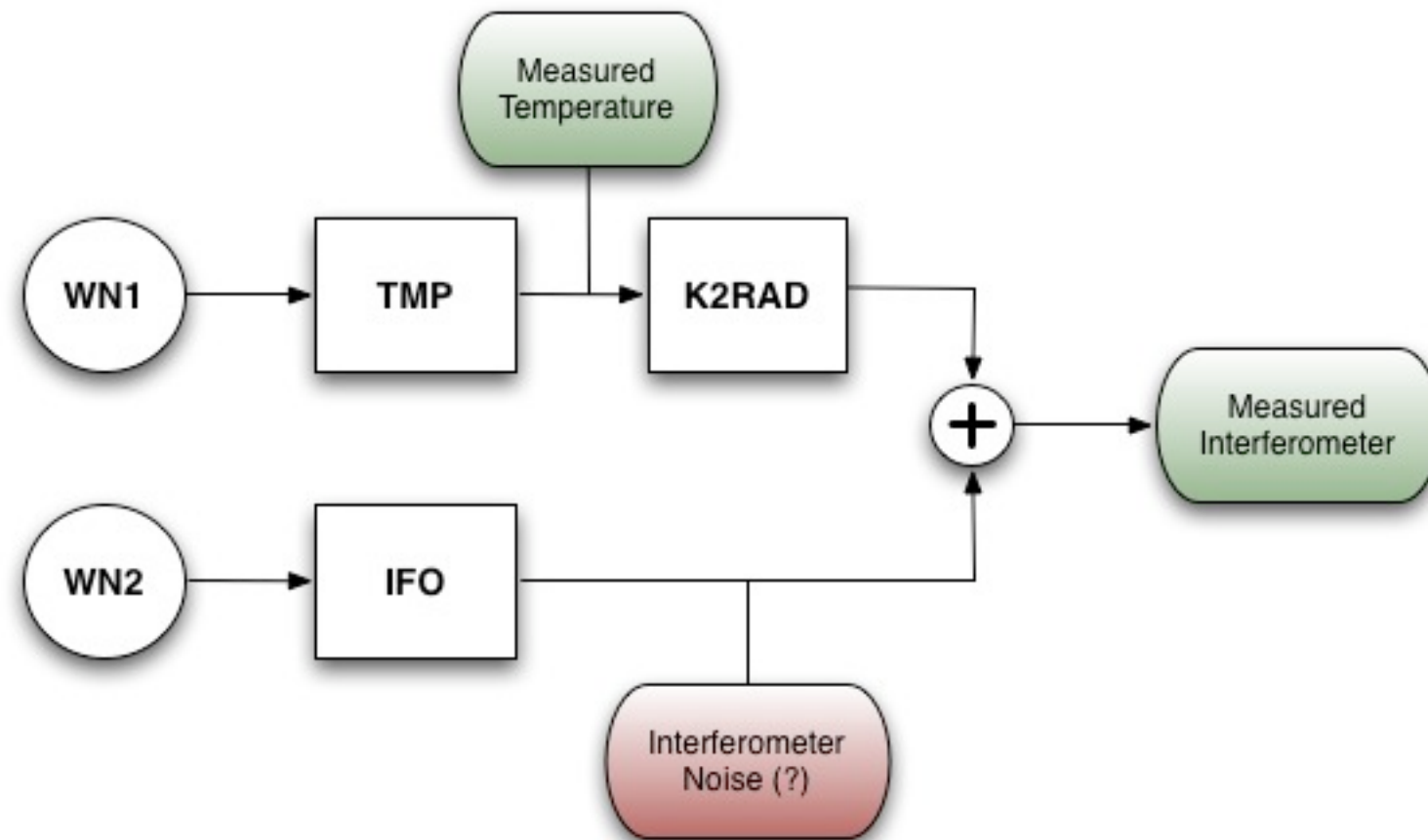
By defining filter properties

- Design a bandpass filter
 - Standard pre-processing step used in LTP lab
 - Alternative to detrending
- Syntax:

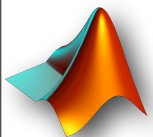

```
Gd = miir(plist('fs',32.47, 'order',...));
```
- Working example: Band pass



IFO/Temperature Example



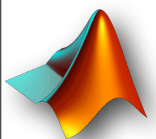
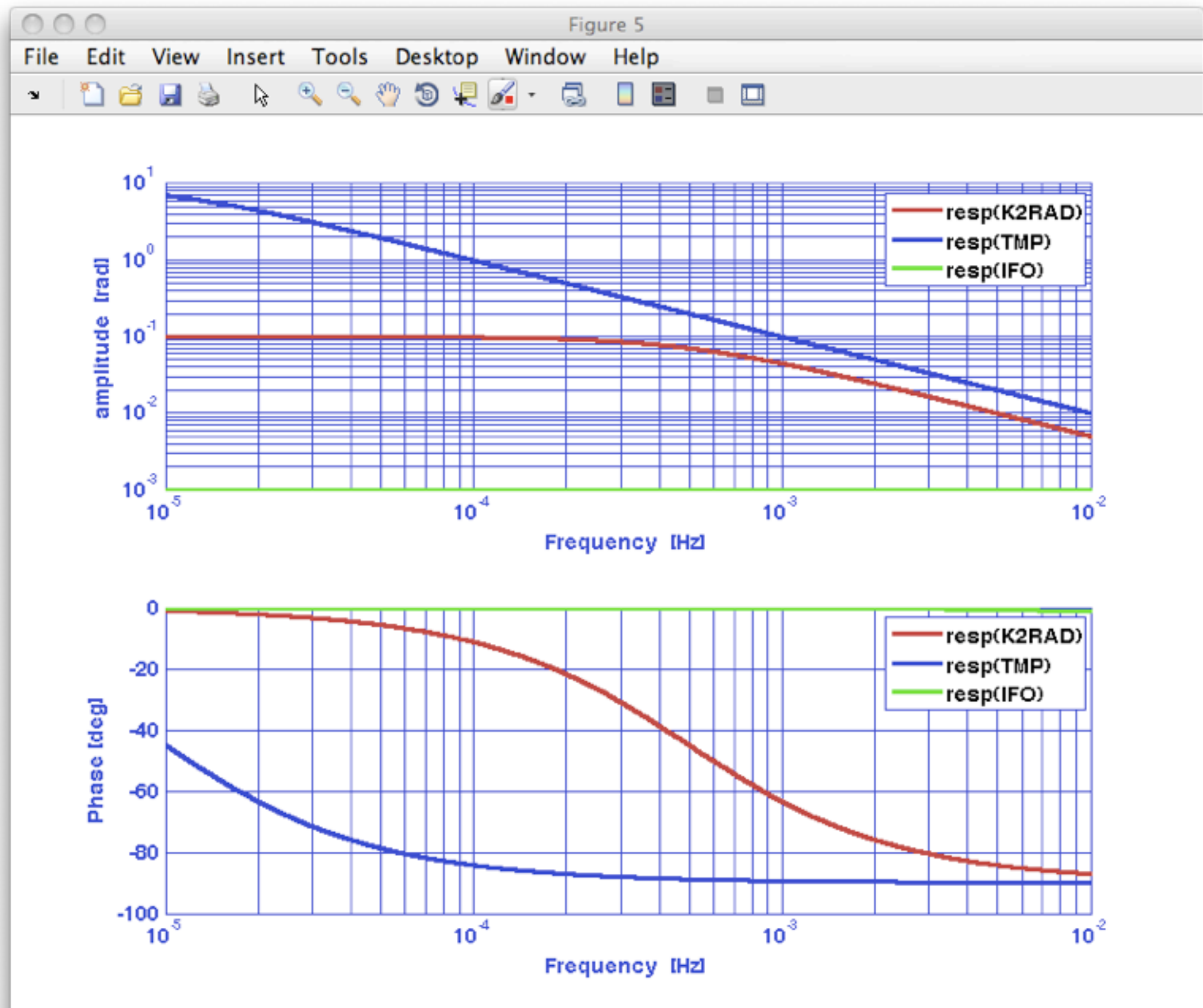
- Aim: perform the analysis with a toy model
 - Create transfer function models: TMP, IFO, K2RAD
 - Discretize
 - Filter (white noise) data
 - Estimate transfer function (topic 3) with synthetic data



IFO/Temperature Example

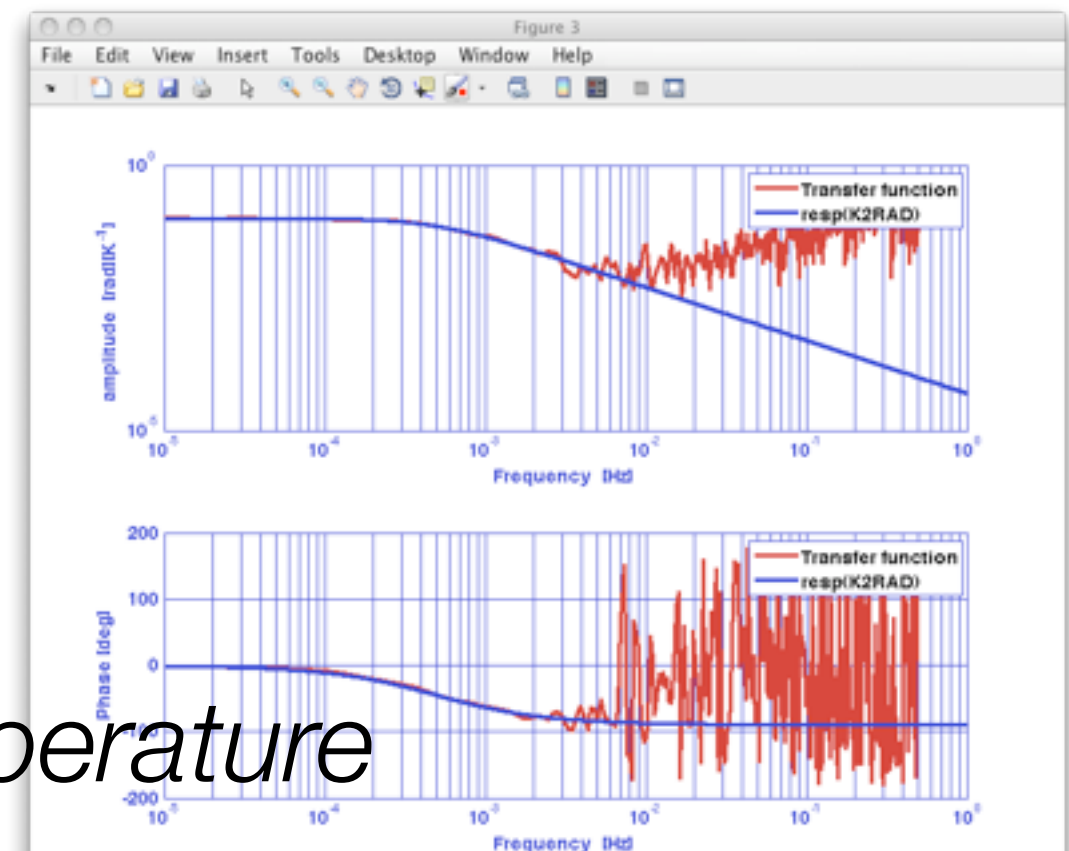
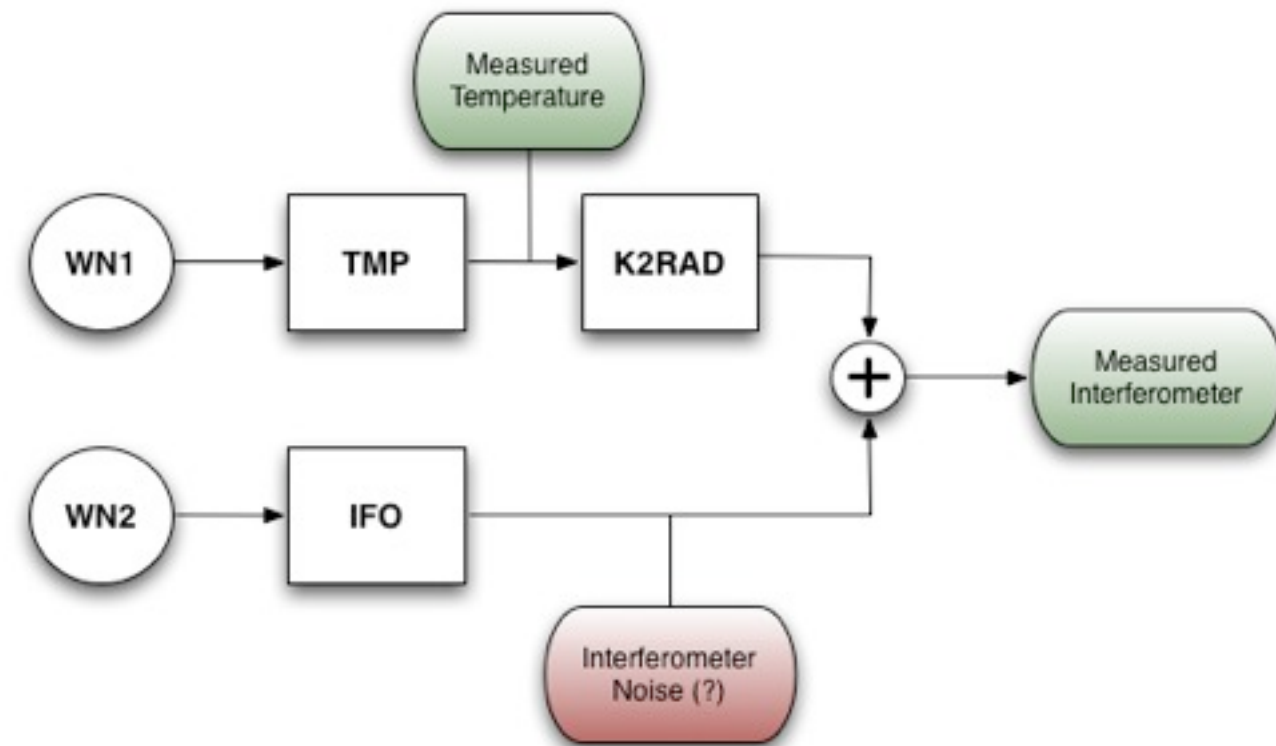


- The toy models

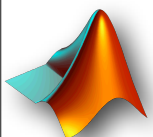


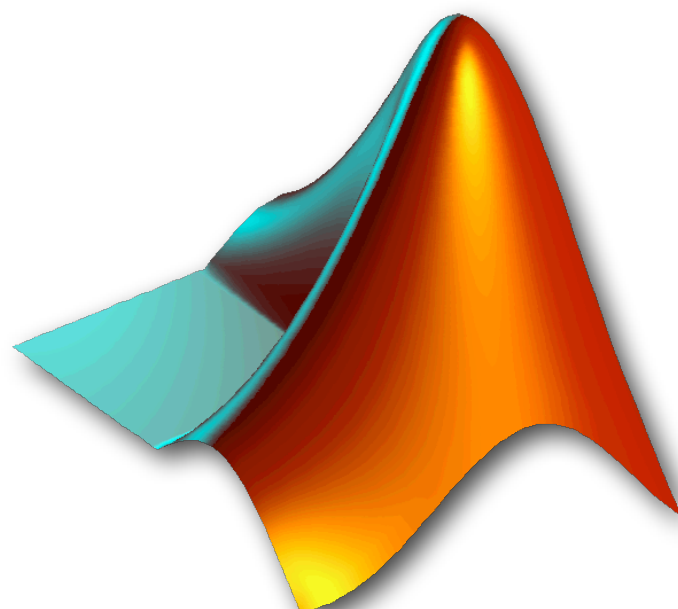
IFO/Temperature Example

- Step-by-step
 - Generate models:
 - TMP, IFO, K2RAD
 - Discretize
 - Build two white-noise time-series
 - Filter with the digital filters
 - Estimate transfer function
 - Project temperature noise



Working example: IFO/Temperature





Topic 5: Fitting to data

- let's see if we will have time...

